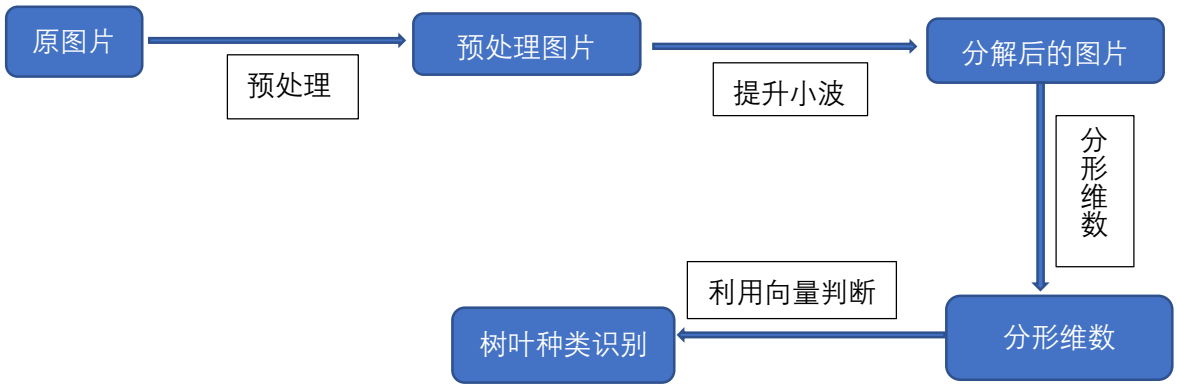
期末大作业程序报告

一，实验对象

分形，根据附上的 pdf 的算法描述，基于以下过程进行不同种类树叶的识别:



更详细的，其中:

1. 预处理：把照片的尺寸调制至一致，这里没有用程序处理，因为该过程不能直接拉伸，否则会影响原有的长宽比，应该选用裁剪，裁剪选取包含单片树叶，不包含文字的部分；此后用程序将 RGB 图片转换为灰度图片。
2. 提升小波：每次提升小波将原图片变为 horizontal detail images(h), vertical detail images(v), diagonal detail images(d) and coarse images(c)，按照原论文的论述，做两次小波后，识别正确率就已足够
3. 分形维数：按照原论文提供的 blanket method 计算分形维数，但原论文

$$u_\varepsilon(i,j) = \max\{u_{\varepsilon-1}(i,j)+1, \max u_{\varepsilon-1}(m,n) \mid d \leqslant 1\}$$
$$b_\varepsilon(i,j) = \max\{b_{\varepsilon-1}(i,j)-1, \max u_{\varepsilon-1}(m,n) \mid d \leqslant 1\}$$

中b_ε的计算公式应有误，宜将b_ε的计算公式两处的 max 改为 min。

4. 利用向量判断：原论文中利用人工神经网络（BPANN）配合大量样本训练得到模型，由于知识和样本数量的限制（事实上，对特定种类的树叶，网上并不能搜到大量单个树叶的平铺图，例如百度图片“枫叶”或者“枫叶平铺图”都不能得到原文要求的图片数量，即 40 张），自然的，我们想通过向量内积来判断，只因为，向量在定义内积后，可以定义角度，自然的定义了相似度，但:

我们这里的向量分量表示的是分形维数，对分形维数，其对加法不成群，若否，则 2 维+2 维=2 维，2 维=0，其他维数同理，也就是说，定义在这个向量上定义内积并没有意义（不可能有线性性），我们没有取用了一个并完全合定义的“内积”:

$$x \cdot y = (x-2.5)(x-2.5)$$

（若定义$x \cdot y = xy$每个任意两个向量的$\frac{x \cdot y}{\sqrt{(x \cdot x)(y \cdot y)}}$都大于 0.9989）

在这种定义下，$\frac{x \cdot y}{\sqrt{(x \cdot x)(y \cdot y)}}$也确实能一定程度的表示相似性，

或者不采用内积定义相似性，考虑差异值

$$different(x,y) = -|x-y|$$

差异值越大，向量值越不相同

赋权时候假设 horizontal detail images(h), vertical detail images(v), diagonal detail images(d) and coarse images(c)各占 1/4 的细节

因此对 h,v,d,ch,cv,cd,cc 各赋权 1/4,1/4,1/4,1/16,1/16,1/16,1/16

二．实验结果

第一部分：向量值

2.0377　　2.0798　　2.1550　　2.1376　　2.2710　　2.4915　　2.7471　　0

2.0452	2.1775	2.1361	2.2262	2.4281	2.3558	2.5250	0
2.0314	2.0946	2.0877	2.1361	2.2874	2.2346	2.4876	0
2.0389	2.1294	2.1504	2.1859	2.3606	2.4282	2.6109	0
2.1327	2.2894	2.2906	2.4646	2.5709	2.5620	2.8271	0
2.1226	2.3020	2.2819	2.3844	2.4999	2.4987	2.5889	0
2.1526	2.3020	2.2694	2.4034	2.5359	2.5062	2.8063	0
2.0806	2.3217	2.2838	2.3589	2.6259	2.6084	2.9289	0

其中，前四个为银杏的向量值，后四个为枫叶的向量值，定型的看，前四个两两比较像，后四个两两也比较像

第二部分

第一种“差异值”

0	-0.0687	-0.0555	-0.0349	-0.1586	-0.1485	-0.1506	-0.1581
-0.0687	0	-0.0606	-0.0338	-0.1441	-0.1142	-0.1361	-0.1436
-0.0555	-0.0606	0	-0.0538	-0.2047	-0.1748	-0.1967	-0.2041
-0.0349	-0.0338	-0.0538	0	-0.1509	-0.1238	-0.1429	-0.1504
-0.1586	-0.1441	-0.2047	-0.1509	0	-0.0362	-0.0242	-0.0421
-0.1485	-0.1142	-0.1748	-0.1238	-0.0362	0	-0.0281	-0.0535
-0.1506	-0.1361	-0.1967	-0.1429	-0.0242	-0.0281	0	-0.0490
-0.1581	-0.1436	-0.2041	-0.1504	-0.0421	-0.0535	-0.0490	0

可以看到同种树叶的相似度较异种树叶的差异值较低（红色值普遍低于黑色值，红色值在 0-0.07，黑色值在 0.11-0.21）

第二种“内积”

1.0000	0.9710	0.9685	0.9918	0.9217	0.9538	0.9446	0.8889
0.9710	1.0000	0.9904	0.9933	0.9076	0.9754	0.9361	0.8706
0.9685	0.9904	1.0000	0.9866	0.8576	0.9403	0.8953	0.8107
0.9918	0.9933	0.9866	1.0000	0.9202	0.9730	0.9453	0.8846
0.9217	0.9076	0.8576	0.9202	1.0000	0.9628	0.9940	0.9870
0.9538	0.9754	0.9403	0.9730	0.9628	1.0000	0.9720	0.9397
0.9446	0.9361	0.8953	0.9453	0.9940	0.9720	1.0000	0.9821
0.8889	0.8706	0.8107	0.8846	0.9870	0.9397	0.9821	1.0000

可以看到同种树叶的相似度较异种树叶的相似度均高（红色值普遍高于黑色值）

三．结果分析

同种树叶经处理后的向量的相似度显著高于异种树叶经处理后的向量的相似度；

同种树叶经处理后的差异值显著低于异种树叶经处理后的差异值。

可以有效的识别树叶。

四．可运行代码

```
A=zeros(8,8);
A(1,1:1:7)=Feature_extraction('1-1.jpg');
A(2,1:1:7)=Feature_extraction('1-2.jpg');
A(3,1:1:7)=Feature_extraction('1-3.jpg');
A(4,1:1:7)=Feature_extraction('1-4.jpg');
A(5,1:1:7)=Feature_extraction('2-1.jpg');
A(6,1:1:7)=Feature_extraction('2-2.jpg');
A(7,1:1:7)=Feature_extraction('2-3.jpg');
A(8,1:1:7)=Feature_extraction('2-4.jpg');
%前四个为银杏的向量值，后四个为枫叶的向量值
%几张图的分辨率都调到了 1024*1024.据论文要求
```

```
%实际只需要四的倍数就行了，但分辨率最好一致，不然还要添加处理
B=zeros(8);
disp('向量值');
disp(A);
for i=1:8
    for j=1:i
        B(i,j)=myint(A(i,1:1:7),A(j,1:1:7));
        B(j,i)=B(i,j);
    end
end
disp('差异值');
disp(B);

function a=Feature_extraction(x)
    y=perprocessing(x);
    [d,h,v,cd,ch,cv,cc]=wavelet_decompose(y);
    a1=fractal_dimension(d);
    a2=fractal_dimension(h);
    a3=fractal_dimension(v);
    a4=fractal_dimension(cd);
    a5=fractal_dimension(ch);
    a6=fractal_dimension(cv);
    a7=fractal_dimension(cc);
    a=[a1 a2 a3 a4 a5 a6 a7];
end

function w=perprocessing(x)
    w=double(imread(x));
    w=rgbtogray(w);
    %可以把 w 再转化为 uint8，以显示，但没必要
end

function y=rgbtogray(x)
    a=size(x,1);
    b=size(x,2);
    y=zeros(a,b);
    for i=1:a
        for j=1:b
            y(i,j)=x(i,j,1)*0.2989+x(i,j,2)*0.5870+x(i,j,3)*0.1141;
            %有些地方用的是 0.1140
        end
    end
end
```

```
function [d,h,v,cd,ch,cv,cc]=wavelet_decompose(y)
    arguments
        y (1024,1024)
    end
    [d,h,v,c]=lifting_wavelet(y);
    [cd,ch,cv,cc]=lifting_wavelet(c);
    %按照原论文的论述，做两次小波后，识别正确率就已足够
end

function [d,h,v,c]=lifting_wavelet(y)
    a=size(y,1);
    b=zeros(a);
    for i=1:a
        b(i,1:1:end)=[y(i,1:2:a),y(i,2:2:a)];%懒变换
        b(i,a/2+1:1:a)=b(i,a/2+1:1:a)-
(b(i,1:1:a/2)+[b(i,2:1:a/2),b(i,a/2)])/2;%预测
        b(i,1:1:a/2)=b(i,1:1:a/2)+(b(i,a/2+1:1:a)+[b(i,a/2+2:1:a),b(i,a
)])/4;%更新
    end
    for i=1:a
        b(1:1:end,i)=[b(1:2:a,i);b(2:2:a,i)];%懒变换
        b(a/2+1:1:a,i)=b(a/2+1:1:a,i)-
(b(1:1:a/2,i)+[b(2:1:a/2,i);b(a/2,i)])/2;%预测
        b(1:1:a/2,i)=b(1:1:a/2,i)+(b(a/2+1:1:a,i)+[b(a/2+2:1:a,i);b(a,i
)])/4;%更新
    end
    c=b(1:1:a/2,1:1:a/2);
    d=b(a/2+1:1:a,a/2+1:1:a);
    v=b(a/2+1:1:a,1:1:a/2);
    h=b(1:1:a/2,a/2+1:1:a);
end

function D=fractal_dimension(I)
    J=zeros([size(I),5]);
    a=size(I,1);
    b=size(I,2);
    J(1:1:a,1:1:b,1)=I;
    K=J;
    L=zeros(1,5);
    A=zeros(1,4);
    for i=2:5
        for j=1:a
            for k=1:b
                if j==1
```

```
                    c0=J(j+1,k,i-1);
                elseif j==a
                    c0=J(j-1,k,i-1);
                else
                    c0=[J(j+1,k,i-1),J(j-1,k,i-1)];
                end
                if k==1
                    d0=J(j,k+1,i-1);
                elseif k==b
                    d0=J(j,k-1,i-1);
                else
                    d0=[J(j,k+1,i-1),J(j,k-1,i-1)];
                end
                %这两段用 intersect 函数写更简洁，但不知为何，运行速度太慢
                J(j,k,i)=max([J(j,k,i-1)+1,c0,d0]);
                K(j,k,i)=min([K(j,k,i-1)-1,c0,d0]);
                L(i)=L(i)+J(j,k,i)-K(j,k,i);
            end
        end
     A(i-1)=(L(i)-L(i-1))/2;
    end
    %按照原论文提供的 blanket method 计算分形维数
    D=polyfit(log(1:1:4),log(A),1);
    D=-D(1)+2;
end

function c=myint(a,b)
    arguments
        a (1,7)
        b (1,7)
    end
    d=zeros(1,7);
    for i=1:7
        d(i)=abs(a(i)-b(i));
        %无法定义在这个向量上定义完全合定义的内积
        %分形维数的加法没有意义，内积无线性性
    end
    c=-d*[4 4 4 1 1 1 1]'/16;
    %赋权时候假设
horizontal detail images(h), vertical detail images(v),
    %diagonal detail images(d) and coarse images(c)各占 1/4 的细节
    %因此对 h,v,d,ch,cv,cd,cc 各赋权 1/4,1/4,1/4,1/16,1/16,1/16,1/16
end
```

“内积”的代码，先运行前一个程序，后运行（可自行修改 myint 改变定义的内积函数）

```
for i=1:8
    A(i,8)=sqrt(myint(A(i,1:1:7),A(i,1:1:7)));
    for j=1:i
        B(i,j)=myint(A(i,1:1:7),A(j,1:1:7))/(A(i,8)*A(j,8));
        B(j,i)=B(i,j);
    end
end
disp('相似度');
disp(B);

function c=myint(a,b)
    arguments
        a (1,7)
        b (1,7)
    end
    d=zeros(1,7);
    for i=1:7
        d(i)=(a(i)-2.5)*(b(i)-2.5);
        %可改成 d(i)=(a(i))*(b(i));
        %或者自己定义的“内积”
    end
    c=d*[4 4 4 1 1 1 1]'/16;
end
```