数字图像处理 project

Project title:	Generating Pattern Classes	Project number:	Project 1201
姓　　名:	彭思达	学　　号:	3140103545
学　　院:	信电学院	专　　业:	信息工程
Date due:	2017.6.30	Date handed in:	June 24, 2017

Abstract

首先文章讨论了边缘追踪算法，使用边缘追踪算法生成了两个图像的边缘。

然后文章讨论了傅立叶描述子，通过减少傅立叶描述子的数量为基础重建边界的效果。

通过对两个傅立叶描述子的向量添加高斯噪声来产生两个模式类，生成两个模式类的训练集和测试集。

June 24, 2017

1 Technical discussion

1.1 对边缘追踪算法的讨论

给定一个二值区域 R 或其边界，追踪 R 的边界或给定边界的算法由如下步骤组成：

- 1. 令起始点 b_0 为图像中左上角标记为 1 的点。使用 c_0 表示 b_0 西侧的邻点。很明显，c_0 总是背景点，从 c_0 开始按顺时针方向参考 b_0 的 8 个邻点。令 b_1 表示所遇到的值为 1 的第一个邻点，并直接令 c_1 是序列中 b_1 之前的点。存储 b_0 和 b_1 的位置，以便在步骤 5 中使用。
- 2. 令 $b = b_1$ 和 $c = c_1$。
- 3. 从 c 开始按顺时针方向行进，令 b 的 8 个邻点为 $n_1 \square n_2 \square n_3 \square \cdots \square n_8$，找到标为 1 的第一个 n_k。
- 4. 令 $b = n_k$ 和 $c = n_{k-1}$。
- 5. 重复步骤 3 和步骤 4，直到 $b = b_0$ 且找到的下一个边界点为 b_1。

当算法停止时，所找到的 b 点的序列就构成了排列后的边界点的集合。

1.2 对傅立叶描绘子的讨论

当我们有一幅图像，我们将每个坐标对当作一个复述来处理，即

$$s(k) = x(k) + jy(k) \tag{1}$$

这种表示方法将二维问题简化成了一维问题。

s(k) 的离散傅立叶变换为

$$a(u) = \sum_{k=0}^{K-1} s(k) e^{-j2\pi uk/K} \tag{2}$$

复系数 a(u) 称为边界的傅立叶描绘子。这些系数的傅立叶反变换可恢复 s(k)。如下所示：

$$s(k) = \frac{1}{K} \sum_{u=0}^{K-1} a(u) e^{j2\pi uk/K} \tag{3}$$

假设仅使用前 P 个傅立叶系数而不使用所有的系数。这等同于令 $a(u) = 0, u \geq P$，结果为 s(k) 的如下近似：

$$\widehat{s}(k) = \frac{1}{K} \sum_{u=0}^{P-1} a(u) e^{j2\pi uk/P} \tag{4}$$

当 P 越小时，边界丢失的细节就越多。

2　Discussion of results

对于边界追踪算法，当算法停止时，所找到的 b 点的序列就构成了排列后的边界点的集合。

对于傅立叶描绘子，我们可以使用前 P 个傅立叶系数进行重建。当 P 越小时，边界丢失的细节就越多。

3　Results

3.1　得到两幅图的边界

通过边界追踪算法得到两张图的边界，Fig. 12.18(a1) 的处理结果如下所示，左边是原图，右边是边界图：

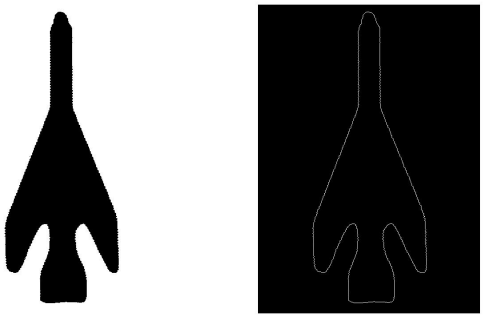
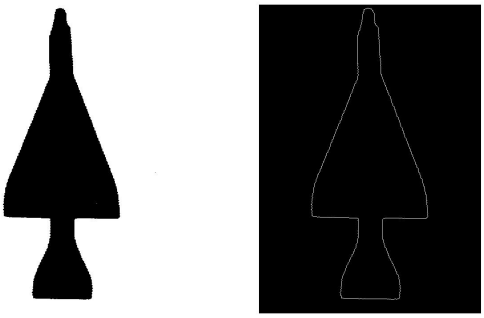


Fig. 12.18(a2) 的处理结果如下所示：



3.2　选择最小数量的傅立叶描绘子

选择最小数量的傅立叶描绘子来保持两幅图的基本差别，Fig. 12.18(a1) 的处理结果如下所示，从左到右、从上到下选择傅立叶描绘子的数量是 100、30、12、10：

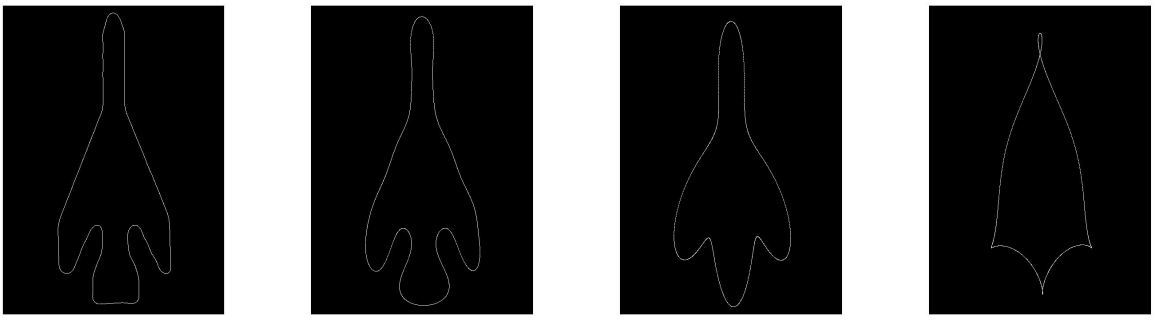
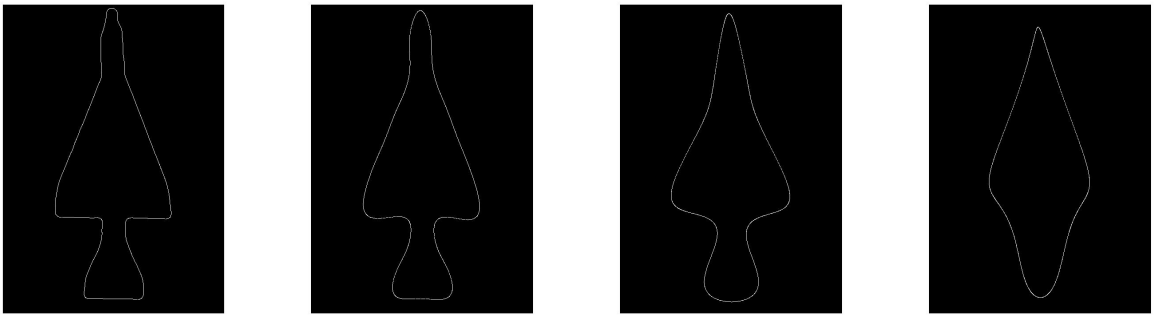


Fig. 12.18(a2) 的处理结果如下所示：



所以可以保持两幅图的基本差别的傅立叶描绘子的数量为 12。

Fig. 12.18(a1) 的描绘子向量为：

```
>> x1

x1 =

   1.0e+05 *

  Columns 1 through 5

   8.3116 + 5.1092i   0.2084 - 3.5080i   0.6280 + 0.0521i  -0.0304 + 0.1093i
       0.1543 + 0.0322i

  Columns 6 through 10

   0.1592 - 0.5287i  -0.0173 + 0.0010i   0.0734 + 0.2402i   0.2077 - 0.0478i
      -0.0820 - 0.4768i

  Columns 11 through 12

   0.8726 - 0.0775i   0.0487 + 1.5551i
```

Fig. 12.18(a2) 的描绘子向量为：

```
>> x2

x2 =

   1.0e+05 *

  Columns 1 through 5

   6.4056 + 4.2775i   0.5042 - 3.0793i   0.1804 + 0.0696i  -0.0832 + 0.1634i
       0.1132 + 0.0669i

  Columns 6 through 10

   0.2450 - 0.2375i   0.1046 - 0.1449i  -0.0424 - 0.0393i   0.0008 - 0.0102i
      -0.2025 - 0.3986i

  Columns 11 through 12

   0.2529 - 0.0869i   0.2188 + 1.5013i
```

3.3　产生两个模式类

模式类 A 的测试集，这里因为数据过多，只取一个向量查看：

```
>> test1(1,:)

ans =

   1.0e+05 *

  Columns 1 through 5

   9.1272 + 4.9787i   0.2144 - 1.9406i  -0.7049 - 1.1672i  -1.4223 + 0.0035i
      -3.4408 + 0.8239i

  Columns 6 through 10

   0.6139 - 0.8374i  -1.2287 + 0.1985i   0.7148 + 0.3745i   1.0693 - 1.4356i
      -0.0409 - 0.3407i

  Columns 11 through 12

   0.6377 - 1.8427i  -1.3411 + 2.6586i
```

模式类 B 的测试集，这里因为数据过多，只取一个向量查看：

```
>> test2(1,:)

ans =

   1.0e+05 *

  Columns 1 through 5

```

```
    6.8714 + 4.1087i   1.1823 - 3.6134i   0.4897 + 0.3990i  -1.0987 - 0.5277i
      -0.3334 - 0.9414i

  Columns 6 through 10

    0.6599 - 0.3688i  -0.1426 - 0.4223i  -0.1154 - 1.0856i  -0.2921 - 0.3634i
      0.4707 + 0.1631i

  Columns 11 through 12

    0.2806 + 0.2161i  -0.4717 + 2.1802i
```

模式类 A 的训练集，这里因为数据过多，只取一个向量查看：

```
>> train1(1,:)

ans =

   1.0e+05 *

  Columns 1 through 5

    5.9851 + 5.2875i   0.0126 - 2.8507i   1.8285 + 1.0901i   0.6061 + 1.4559i
      1.0746 - 0.1723i

  Columns 6 through 10

   -1.4687 - 1.0277i   0.8571 - 0.0334i  -1.3252 + 1.3824i   0.1166 - 0.4462i
     -0.0563 - 0.8955i

  Columns 11 through 12

    0.6478 + 0.8912i   0.6946 + 3.3497i
```

模式类 B 的训练集，这里因为数据过多，只取一个向量查看：

```
>> train2(1,:)

ans =

   1.0e+05 *

  Columns 1 through 5

    5.4404 + 4.9561i   0.9894 - 2.7216i   0.5606 + 0.6833i   0.8738 + 0.6487i
      -0.6565 + 0.1561i

  Columns 6 through 10

    0.7184 - 0.3297i  -0.0728 - 0.2335i  -0.1924 + 0.6477i   1.6704 - 0.5661i
      0.3425 + 0.4578i

  Columns 11 through 12

  -0.6965 + 0.7526i   0.6203 + 0.5151i
```

4 Appendix

总的测试代码如下：

```
% test.m
img = imread('Fig1218(airplanes).tif');

% 提取图像中的a1部分，提取图像中的a2部分
[img1, img2, img3, img4] = getImage(img);

figure;
imshow(img1);
figure;
imshow(img2);

% 得到图像a1边界图像，得到图像a2边界图像
[boundary1, boundary2, img5, img6] = boundary(img1, img2,img3, img4);

figure;
imshow(boundary1);
figure;
imshow(boundary2);

P = 12;

% 利用图像a1的傅立叶描绘子重建边界，利用图像a2的傅立叶描绘子重建边界
[boundary3, boundary4, x1, x2] = fourier(img1, img2, img5, img6, P);

figure;
imshow(boundary3);
figure;
imshow(boundary4);

% 获取两类模式的训练集的实现，获取两类模式的测试集的实现
[train1, train2, test1, test2] = noise(x1, x2, P);
```

提取图像中特定部分的代码如下：

```
% getImage.m
function [ img1, img2, img3, img4 ] = getImage( img )

    img1 = zeros(695, 500);
    img2 = zeros(695, 500);
    img3 = zeros(695, 500);
    img4 = zeros(695, 500);

    for m=1:695                         % 提取图像中的a1部分
        for n=1:500
            img1(m,n) = img(m+80,n+15);
            img3(m,n) = ~img(m+80,n+15);
        end
    end

    for m=1:695                         % 提取图像中的a2部分
        for n=1:500
            img2(m,n) = img(m+80,n+690);
            img4(m,n) = ~img(m+80,n+690);
```

```
        end
    end

end
```

得到图像边界的实现代码如下：

```
% boundary.m
function [ g1, g2, i5, i6 ] = boundary( img1, img2, img3, img4 )

    [M1,N1] = size(img1);                    % 获取图像a1的行数和列数
    [M2,N2] = size(img2);                    % 获取图像a2的行数和列数

    [B1, ~, ~, ~] = bwboundaries(img3);      % 获取图像a1边界点坐标
    [B2, ~, ~, ~] = bwboundaries(img4);      % 获取图像a2边界点坐标

    i5=B1{1,1};
    n1=size(i5,1);                       % 获取图像a1边界点数量
    g1=zeros(M1,N1);

    for n=1:n1                           % 得到图像a1边界图像
     g1(i5(n,1),i5(n,2))=1;
    end

    i6=B2{1,1};
    n2=size(i6,1);                       % 获取图像a2边界点数量
    g2=zeros(M2,N2);

    for n=1:n2                           % 得到图像a2边界图像
     g2(i6(n,1),i6(n,2))=1;
    end

end
```

利用图像的傅立叶描绘子重建边界的实现代码如下：

```
% fourier.m
function [ g3, g4, x1, x2 ] = fourier( i1, i2, i5, i6, P )

    [M1,N1] = size(i1);                  % 获取图像a1的行数和列数
    [M2,N2] = size(i2);                  % 获取图像a2的行数和列数
    n1 = size(i5,1);
    n2 = size(i6,1);

    if P >= n1
        P = n1;
    end

    n3 = (n1-P)/2;

    a1 = zeros(1, n1);
    s = zeros(1, n1);

    for n = 0:n1-1                       % 获取图像a1的傅立叶描绘子
        for m = 0:n1-1
            a1(n+1) = a1(n+1) + (i5(n1-m,1) + j*i5(n1-m,2)) * exp(-j*2*pi*n*m/n1);
        end
    end

```

```
for n = 0:n1-1                              % 利用图像a1的傅立叶描绘子重建边界
    for m = 0:(1006-n3)
        s(n+1) = s(n+1)+1/n1*a1(m+1) * exp(j*2*pi*m*n/n1);
    end

    for m = (1007+n3):n1-1
        s(n+1) = s(n+1)+1/n1*a1(m+1) * exp(j*2*pi*m*n/n1);
    end
end

g3 = zeros(M1,N1);
s1 = zeros(1, n1);
s2 = zeros(1, n1);
for n = 1:n1
    s1(n) = real(s(n));                     % 获取重建图像a1边界得到复数的实部
    s2(n) = imag(s(n));                     % 获取重建图像a1边界得到复数的虚部
    g3(round(real(s(n))),round(imag(s(n))))=1;    % 得到重建图像a1边界图像
end

x1 = zeros(1, P);
for n=0:(1006-n3)                         % 获取用于重建图像a1边界的傅立叶描绘子
    x1(n+1) = a1(n+1);
end

for n = (1007+n3):n1-1
    x1(n-n1+P+1) = a1(n+1);
end

n4 = (n2-1-P)/2;
a2 = zeros(1,n2);
s = zeros(1,n2);

for n = 0:n2-1                              % 获取图像a2的傅立叶描绘子
   for m = 0:n2-1
    a2(n+1) = a2(n+1) + (i6(n2-m,1)+j*i6(n2-m,2)) * exp(-j*2*pi*n*m/n2);
    end
end

for n = 0:n2-1                              % 利用图像a2的傅立叶描绘子重建边界
    for m = 0:(854-n4)
        s(n+1) = s(n+1)+1/n2*a2(m+1) * exp(j*2*pi*m*n/n2);
    end
    for m = (856+n4):n2-1
        s(n+1) = s(n+1)+1/n2*a2(m+1) * exp(j*2*pi*m*n/n2);
    end
end

g4 = zeros(M2,N2);
for n = 1:n2
    s1(n) = real(s(n));                     % 获取重建图像a2边界得到复数的实部
    s2(n) = imag(s(n));                     % 获取重建图像a2边界得到复数的虚部
    g4(round(real(s(n))),round(imag(s(n)))) = 1;    % 得到重建图像a2边界图像
end

x2 = zeros(1, P);
for n = 0:(854-n4)                          % 获取用于重建图像a2边界的傅立叶描绘子
    x2(n+1) = a2(n+1);
end

for n = (856+n4):n2-1
```

```
84          x2(n-n2+P+1) = a2(n+1);
85      end
86
87
88  end
```

获取两类模式的训练集, 获取两类模式的测试集的代码如下：

```
1   noise.m
2   function [ x3, x4, x5, x6 ] = noise( x1, x2, P )
3
4       m1 = max(abs(x1))/10;                   % 获取加入重建图像a1边界的傅立叶描绘子的
            高斯噪声的方差的值
5       m2 = max(abs(x2))/10;                  % 获取加入重建图像a2边界的傅立叶描绘子的高
            斯噪声的方差的值
6
7       [~, N] = size(x1);
8       x3 = zeros(100, N);
9       x4 = zeros(100, N);
10      for n = 1:100                           % 获取两类模式的训练集的实现
11          x3(n,:) = x1 + normrnd(0,m1,1,P) + j*normrnd(0,m1,1,P);
12          x4(n,:) = x2 + normrnd(0,m2,1,P) + j*normrnd(0,m2,1,P);
13      end
14
15      x5 = zeros(100, N);
16      x6 = zeros(100, N);
17      for n=1:100                            % 获取两类模式的测试集的实现
18          x5(n,:) = x1 + normrnd(0,m1,1,P) + j*normrnd(0,m1,1,P);
19          x6(n,:) = x2 + normrnd(0,m2,1,P) + j*normrnd(0,m2,1,P);
20      end
21
22  end
```