数字图像处理 project

Project title:	Boundary Extraction	Project number:	Project 0502
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:	May 9, 2017

Abstract

这篇报告实现了边界提取算法。首先文章讨论了图片的腐蚀操作，给出了它的数学表示，并使用代码实现了腐蚀操作。

随后文章又讨论了两张图片的相减，给出了它的数学表示，并用代码实现了相减操作。

最后文章讨论了边界提取算法，通过结合腐蚀操作和相减操作，实现了边界提取算法。

文章以书上 fig 9.14(a) 为实验对象，得到了它原图像、腐蚀图像以及边界提取以后得到的边界，并在文章的最后附上了相应的实现代码。

May 9, 2017

1　Technical discussion

1.1　对腐蚀操作的讨论

作为 Z^2 中的集合 A 和 B，表示为 $A \ominus B$ 的 B 对 A 的腐蚀义为：

$$A \ominus B = \{z|(B)_z \subseteq A\} \tag{1}$$

表面上，该式指出了 B 对 A 的腐蚀是一个用 z 平移的 B 包含在 A 中的所有的点 z 的集合。在下面的讨论中，假定集合 B 是一个结构元。因为 B 必须包含在 A 中这一陈述等价于 B 不与背景共享任何公共元素，故我们可以将腐蚀表达为如下的等价形式：

$$A \ominus B = \{z|(B)_z \cap A^c = \emptyset\} \tag{2}$$

上式中，A^c 是 A 的补集，$\emptyset$ 是空集。

1.2　对相减操作的讨论

相减操作就是两张图片对应的元素值相减，这里黑的减黑的仍然是黑的，白的减白的变成黑的，黑的减白的仍然是白的，白的减黑的仍然是黑的。而黑的像素值为 0，白的像素值为 1，所以实现相减操作时需要做相应处理。公式如下：

$$A - B \tag{3}$$

1.3　对边界提取的讨论

表示为 $\beta(A)$ 的集合 A 的边界可以通过先用 B 对 A 腐蚀，而后执行 A 和腐蚀的结果之间的集合之差得到，即

$$\beta(A) = A - (A \ominus B) \tag{4}$$

其中 B 是一个适当的结构元。

2　Discussion of results

腐蚀操作公式如下：

$$A \ominus B = \{z|(B)_z \subseteq A\} \tag{5}$$

相减操作公式如下：

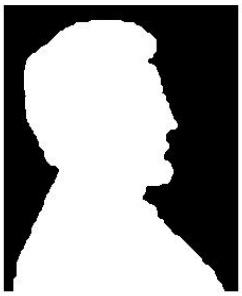
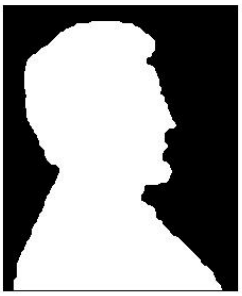
$$A - B \tag{6}$$

边界提取公式如下：

$$\beta(A) = A - (A \ominus B) \tag{7}$$

3 Results

下面三张图像，分别是原图像、腐蚀以后的图像以及图像提取出来的边界。


4 Appendix

实现腐蚀操作的代码如下：

```
% erode.m
function [ output ] = erode( pic, struct )

    [m, n] = size(pic);
    % 填充边界
    pic = fill(pic);
    output = zeros(m, n);

    for i = 1:m
        for j = 1:n
            pic_i = i + 1;
            pic_j = j + 1;
            % 对像素点进行腐蚀操作
            output(i, j) = real_erode(pic(pic_i-1:pic_i+1, pic_j-1:pic_j+1), struct
                );
        end
    end

end

function [ pixel ] = real_erode(region, struct)
    % 只有结构元的阴影部分与图像区域全部相等时，返回像素才会为阴影区域1
    for i = 1:3
        for j = 1:3
            if struct(i, j) == 1 && region(i, j) ~= 1
```

```
                pixel = 0;
                return;
            end
        end
    end
    pixel = 1;
end

function [ pic ] = fill( pic )
    [m, n] = size(pic);
    pic = [zeros(1, n); pic; zeros(1, n)];
    pic = [zeros(m+2, 1), pic, zeros(m+2, 1)];
end
```

相减操作的实现代码如下：

```
% difference.m
function [ output ] = difference( A, B )

    output = A - B;
    output = logical(output);

end
```

提取边界的实现代码如下：

```
% extract.m
function [ output ] = extract( pic )

    struct = ones(3, 3);

    % 腐蚀图像
    erode_pic = erode(pic, struct);
    % 两幅图像相减
    output = difference(pic, erode_pic);

end
```

测试代码如下：

```
% test.m
img = imread('Fig0914(a)(licoln from penny).tif');
img = double(img);

img = extract(img)

img = logical(img);
imshow(img);
```