上海应用技术大学

计算机科学与信息工程学院

课

题

报

告

课程学期： 2020~2021 第一学期

课程名称： 机器视觉应用

课题名称： 基于 Matlab 语言环境 探究 Candy 边缘检测算子原理及应用

学组成员： 刘子健（1910400604）、王恺昊（1910400605）、张珅铖（1910400614）

指导教师： 张晴

一、基础内容摘要：

（一）Candy 算子基本介绍：

Canny 算子是 John.F.Canny 于 20 世纪 80 年代提出的一种多级边缘检测算法。该算子最初的提出是为了能够得到一个最优的边缘检测，即：检测到的边缘要尽可能跟实际的边缘接近，并尽可能的多，同时，要尽量降低噪声对边缘检测的干扰。

Candy 算子作为轮廓线检测中的一个经典算法，它的基本算法思路是在输入的图像中找出一个具有局部最大梯度幅值的像素点。在图像的轮廓检测中还必须满足以下两个条件:必须可以有效的抑制图像的噪声;必须能对轮廓边缘位置精确定位。

Candy 算子核心的原理是使用一个滤波器，它在过滤噪声的同时又可以不改变轮廓线的特性进行边缘检测，实际上 Candy 算子在这个算法中使用一阶微分的滤波器来检测。使用二维高斯函数在任意方向上的一阶导数，把它作为改变输入图的噪声滤波器，通过滤波器与图像卷积和滤波操作，最后对滤波以后的输出图进行寻找图像梯度场的最大变化值，这个值就是图像的轮廓特征代表，也是 Candy 算子的检测方法的结果。

（二）Candy 算子的三个最优准则：

(1)最优检测:对真实边缘不漏检，即要求输出信噪比最大。

(2)最优检测精度:检测的边缘点的位置距实际的边缘点的位置最近。

(3)检测点与边缘点一一对应:每个实际存在的边缘点和检测的边缘点是一对应关系。

（三）Candy 算子的检测算法步骤简述：

步骤 1：用高斯滤波器平滑处理原图像；（图像去噪）

步骤 2：用一阶偏导的有限差分进行计算梯度的幅值和方向；（特征增强）

步骤 3：对梯度幅值进行非极大值抑制；（边缘检测）

步骤 4：用双阈值算法检测和连接边缘。（形态学处理）

二、Candy 算子实现方法及实践：

（零）操作环境及样本选取：

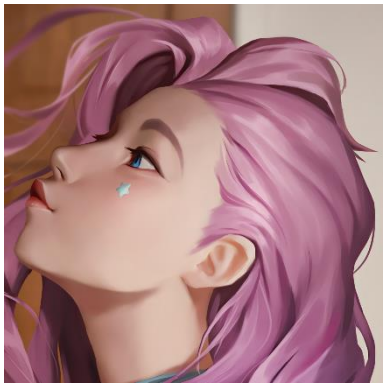
0.1. 实验操作环境：

Windows 10 64 位环境下 Matlab R2020a

0.2. 实验样本：

本实验分别选取了细节内容较为丰富及细节内容较少的图片样本各一张，分辨率分别为 1080*1080 与 1350*1350（如图 0.2.1 及图 0.2.2 所示）

（图 0.2.1 细节内容较为丰富）



（图 0.2.2 细节内容较少）

（一）读入待处理文件及初始化：

1.1. 代码实现部分：

```
img=imread('原图 1080x1080.png');          %读入图像:
[h,w,d]=size(img);                         %获取图像大小：h 行 w 列 d 维
if d>1
    imggray = rgb2gray(img); end           %灰度处理
```

（二）使用高斯滤波器平滑处理原图像：

1.1. 基本原理：

$$h(x,y,\sigma)=\frac{1}{2\pi\sigma^2}exp(-\frac{x^2+y^2}{2\sigma^2})$$

对原图像$f(x,y)$进行高斯平滑处理，得到处理后的图像$g(x,y)$如下：

$$g(x,y)=h(x,y,\sigma)*f(x,y)$$

（注：$*$ 表示卷积）

使用平滑滤波的原因从根本上来说是边缘检测算子中的导数计算。导数计算对噪声十分敏感，如果不提前使用滤波器加以改善，则在导数计算后，噪声将会被放大，使得检测出来的虚假边缘变多，不利于边缘的提取。

平滑滤波和边缘检测是一对矛盾的概念。一方面，平滑滤波能够有效的抑制噪声，而在此过程中会使得图像边缘模糊，增加了边缘定位的不确定性。另一方面，平滑滤波能够除去对边缘检测中导数运算敏感的噪声，有效的抑制了虚假边缘的产生。实际工程经验表明，高斯滤波器可以在抗噪声干扰和边缘检测精确定位之间提供一个较好的折中方案。

1.2. 疑问点：

Q1：canny 算子的步骤中 ，平滑是使用的高斯，为什么选择是高斯？

A1：高斯函数是唯一可分离的圆对称滤波器，所以大多数边缘检测算法都使用它。边缘一般出现在颜色，亮度，或者纹理不一样的区域。

一种方式是将边缘定义为亮度变化剧烈的区域，数学上定义一个表面的

斜率和方向是通过梯度来实现的，而求取图像导数会强调高频率部分而放大了噪声。所以一般在计算梯度之前要进行低通滤波。要使边缘检测器的响应与方向无关，需要使用一个圆对称的平滑滤波器。高斯函数是唯一可分离的圆对称滤波器，所以大多数边缘检测算法都使用它。

1.3. 即将用到的 Matlab 函数详解：

1.3.1. fspecial 函数：

fspecial 函数用于建立预定义的滤波算子，其语法格式为：

h = fspecial(type)

h = fspecial(type，para)

其中 type 指定算子的类型，para 指定相应的参数；

本次使用的 type 类型为：'gaussian'即高斯滤波（Gaussian lowpass filter)：

高斯滤波有两个参数，hsize 表示模板尺寸，默认值为[3 3]，sigma 为滤波器的标准值，单位为像素，默认值为 0.5.

H = FSPECIAL('gaussian',HSIZE,SIGMA) returns a rotationally

本实验所使用的3 × 3高斯滤波如图 1.3.1.1 所示：

	1	2	3
1	0.0751	0.1238	0.0751
2	0.1238	0.2042	0.1238
3	0.0751	0.1238	0.0751

（图 1.3.1.1 本实验所使用高斯滤波）

1.3.2. imfilter 函数：

函数名称：imfilter

函数语法：g=imfilter(f,w, mode,boundary_options,size_optinos)

函数功能：对任意类型数组或多维图像进行滤波

参数介绍：f 是输入图像，w 为滤波模板，g 为滤波结果；表 1-1 总结了其他参数的含义。如图 1.3.2.1 所示为该函数对应参数：

选项		说明
滤波模式 filtering_mode	'corr'	滤波通过使用相关来完成，这是默认值
	'conv'	滤波通过使用卷积来完成
边界选项 boundary_options	P	输入图像的边界通过值 P 填充来扩展；P 的默认值是 0
	'replicate'	图像大小通过复制外边界的值来扩展
	"symmetric'	图像大小通过沿自身的边界进行镜像映射扩展
	'circular'	图像大小通过将图像作为二维周期函数的一个周期来扩展
大小选项 size_optinos	'full'	输出图像的大小与被扩展（填充）图像的大小相同
	'same'	输出图像的大小与输入图像的大小相同，这可通过将滤波模板的中心点的偏移限制为原始图像中包含的点来实现，这是默认值

（图 1.3.2.1 imfilter 函数 函数对应参数）

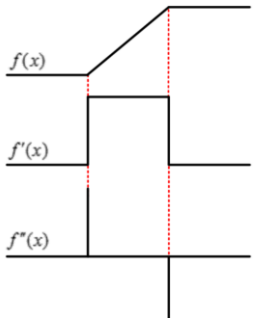
1.4. 代码实现部分：

```
gausFilter = fspecial('gaussian',[3,3],1);                    %生成高斯滤波模板
imgGaus=imfilter(imgGray, gausFilter, 'replicate');    %生成高斯处理后的图像
imgGaus=double(imgGaus);                                      %unit8->double
```

（三）用一阶偏导的有限差分进行计算梯度的幅值和方向：

3.1. 基本原理：

图像的边缘有方向和幅度两个属性，沿边缘方向像素变化平缓，垂直于边缘方向像素变化剧烈，边缘上的这种变化可以用微分算子检测出来，通常用一阶或二阶导数来检测边缘。（如图 3.1.1 为一阶二阶导数的性质）



（一阶二阶导数的性质）

由上图可以看出，一阶导数可以用于检测图像中的一个点是否是边缘点（也就是判断一个点是否在斜坡上）。同样，二阶导数的符号可以用于判断一个边缘像素是否在亮的一边还是暗的一边。用一阶偏导的有限差分来计算梯度的幅值和方向。

3.2. 函数模板：

本实验采用 Sobel 函数进行边缘强度检测：

Sobel 算子的卷积模板为：

a	b	c
d	e	f
g	h	i

-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

Sobel 算子

$$g_x = \frac{\partial f}{\partial x} = g + 2h + i - a - 2b - c$$

$$g_y = \frac{\partial f}{\partial y} = c + 2f + i - a - 2d - g$$

3.3. 边缘强度计算：

$$\sqrt{{g_x}^2 + {g_y}^2}$$

3.4. 确定边缘的方向：

$$\boldsymbol{\theta(x,y) = arctan(\frac{g_x(x,y)}{g_y(x,y)})}$$

3.5. 即将用到的 Matlab 函数详解：

3.5.1. 反正切函数 atand 函数：

语法：Y = atand(X)

Y = atand(X) 返回 X 的元素的反正切 (tan-1)（以度为单位）。该函数同时接受实数和复数输入。

对于 X 的实数值，atand(X) 返回区间 [-90, 90] 中的值。

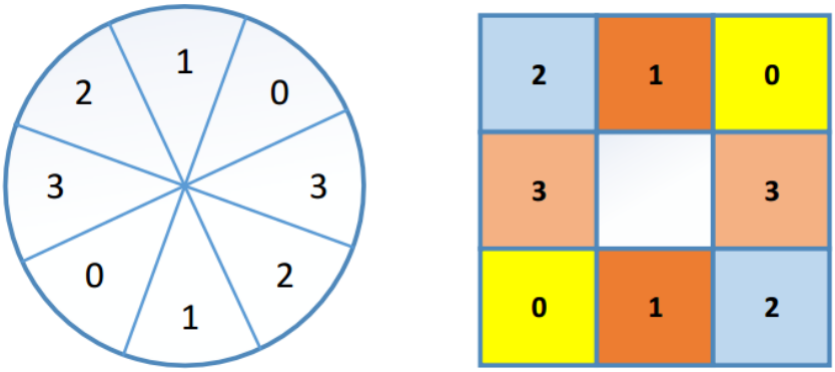
对于 X 的复数值，atand(X) 返回复数值。

3.6. 将方向分为 3 个区域并打标签：

仅仅得到全局的梯度并不足以确定边缘。因此，为了确定边缘，必须保留局部梯度最大的点，且要抑制非极大值。

图像梯度幅值矩阵中的元素值越大，说明图像中该点的梯度值越大，但这并不能说明该点就是边缘(这仅仅是属于图像增强的过程)。在 Canny 算法中，非极大值抑制是进行边缘检测的重要步骤，通俗意义上是指寻找像素点局部最大值，将非极大值点所对应的灰度值置为 0,这样可以剔除掉一大部分非边缘的点。解决这个问题的方法就需要利用梯度的方向,与邻域像素的梯度幅值进行比.较，从而抑制非极大值。

如图 3.6.1 所示为分区图:



（图 3.6.1 分区图）

以待判断像素点为中心分成四个扇区，代表梯度方向近似的四个可能角度$(0°,45°,90°,135°)$见图。将梯度角离散为圆周的四个扇区之一，以便使用3×3的窗口做抑制运算。四个扇区的标号从 0 到 3,对应于3×3的 8 邻域空间四种可能组合。

对每一个像素点的梯度进行非极大值抑制，首先将待判断像素梯度值$M(x,y)$与沿着梯度方向的 2 个 8 邻域像素的梯度$M(x+1,y+1)$进行比较。如果该像素点位置的梯度幅值没有沿梯度方向的两个相邻像素的梯度值大,则说明

该点的梯度值不是局部最大值需被抑制，令$M(x,y)=0$,即该点不是边缘点。用此方法进行抑制全局梯度幅值中的非极大值。

这一步骤将会排除非边缘的像素,仅仅保留了一些细的线条，即就是候选的边缘。

3.7. 代码实现部分：

```
for i=2:h-1
    for j=2:w-1
    %使用 Sobel 函数计算 Gx 与 Gy 以确定:
    Gx=im(i-1,j-1)+2*im(i-1,j)+im(i-1,j+1)-im(i+1,j-1)-2*im(i+1,j)-im(i+1,j+1);
    Gy=im(i-1,j+1)+2*im(i,j+1)+im(i+1,j+1)-im(i-1,j-1)-2*im(i,j-1)-im(i+1,j-1);
    EdgeStrength(i,j)=(Gx^2+Gy^2)^0.5;
    %确定边缘的方向:
    EdgeDie(i,j)=atand(Gx/Gy) ;%atand 1/tan
    %将方向分为 3 个区域并打标签为 1，2，3，0:
    temp=EdgeDie(i,j);
    if (temp<67.5)&&(temp>22.5)
        Sector(i,j)=0;
    elseif (temp<22.5)&&(temp>-22.5)
        Sector(i,j)=3;
    elseif (temp<-22.5)&&(temp>-67.5)
        Sector(i,j)=2;
    else
        Sector(i,j) =1;
    end
    end
end
```

（四）用双阈值算法检测和连接边缘：

4.1. 基本原理：

对非极大值抑制后的图像与高阈值和低阈值进行判断，根据判断的情况分为

以下三种情况:

(a)如果某一像素位置的梯度幅值超过高阈值，则该像素被保留为边缘像素。

(b)如果某一像素位置的梯度幅值小于低阈值，则该像素被排除。.

(c) 如果某一像素位置的梯度幅值介于高、低阈值之间，则判断该像素 8 邻域空间的像素是否存在高于高阈值的像素。如果存在，则该像素将被保留。

4.2. 代码实现部分：

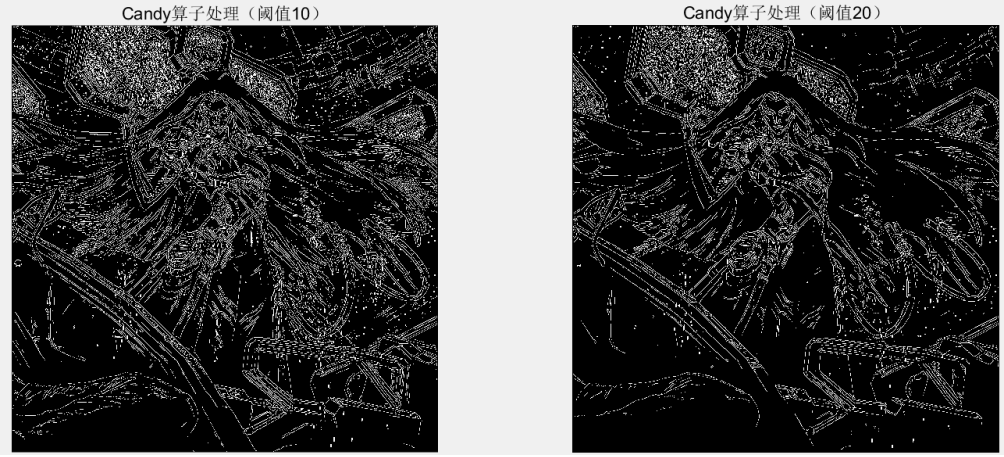
```
%非极大值抑制
for i=2:h-1
    for j=2:w-1
        if 0 == Sector(i,j) %右上-左下
            if (EdgeStrength(i,j)>EdgeStrength(i-1,j+1))&&(EdgeStrength(i,j)>EdgeStrength(i+1,j-1))
                CannyRestrain(i,j)=EdgeStrength(i,j);
            else
                CannyRestrain(i,j)= 0;
            end
        elseif 1 == Sector(i,j) %竖直方向
            if ( EdgeStrength(i,j)>EdgeStrength(i-1,j) )&&( EdgeStrength(i,j)>EdgeStrength(i+1,j)   )
                CannyRestrain(i,j) = EdgeStrength(i,j);
            else
                CannyRestrain(i,j) = 0;
            end
        elseif 2 == Sector(i,j) %左上 - 右下
            if ( EdgeStrength(i,j)>EdgeStrength(i-1,j-1) )&&( EdgeStrength(i,j)>EdgeStrength(i+1,j+1))
                CannyRestrain(i,j) = EdgeStrength(i,j);
            else
                CannyRestrain(i,j) = 0;
            end
        elseif 3 == Sector(i,j) %横方向
            if ( EdgeStrength(i,j)>EdgeStrength(i,j+1) )&&( EdgeStrength(i,j)>EdgeStrength(i,j-1)   )
                CannyRestrain(i,j) = EdgeStrength(i,j);
            else
                CannyRestrain(i,j) = 0;
            end
        end
    end
end

%双阈值检测
ratio = 2;%比
for i=2:h-1
    for j=2:w-1
        if CannyRestrain(i,j)<lowT %低阈值处理
            Canny(i,j) = 0;
            Bina(i,j) = 0;
            continue;
        elseif CannyRestrain(i,j)>ratio*lowT %高阈值处理
            Canny(i,j) = CannyRestrain(i,j);
            Bina(i,j) = 1;
            continue;
        else %介于之间的看其 8 领域有没有高于高阈值的，有则可以为边缘
```

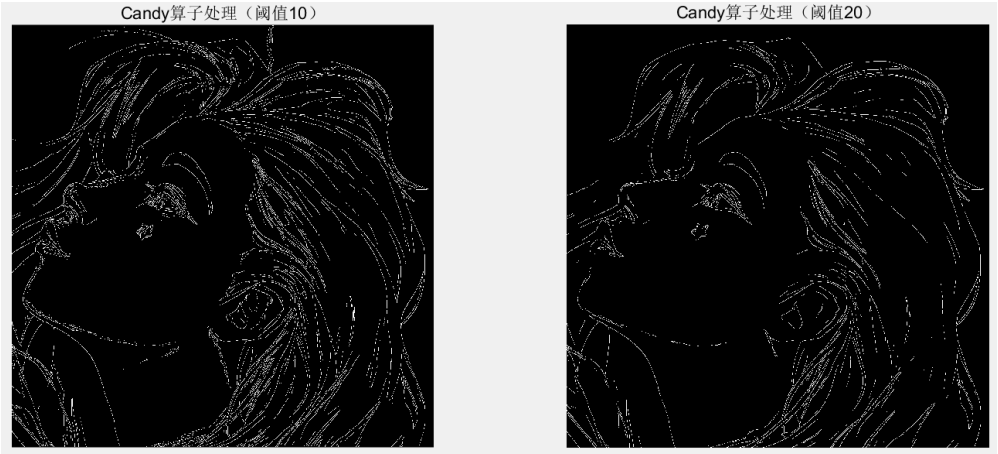
```
            tem =[CannyRestrain(i-1,j-1), CannyRestrain(i-1,j), CannyRestrain(i-1,j+1);
            CannyRestrain(i,j-1),      CannyRestrain(i,j),     CannyRestrain(i,j+1);
            CannyRestrain(i+1,j-1), CannyRestrain(i+1,j), CannyRestrain(i+1,j+1)];
            temMax = max(tem);
            if temMax(1) > ratio*lowT
               Canny(i,j) = temMax(1);
               Bina(i,j) = 1;
               continue;
            else
                Canny(i,j) = 0;
                Bina(i,j) = 0;
                continue;
                 end
            end
        end
end
```

三、Candy 算子处理结果：

对于样本 1（细节内容较为丰富）：



对于样本 2（细节内容较少）：



参考文献:

[1] 孙腾云.基于神经网络的边缘检测 [D].西安电子科技大学.2007:15-18
[2] 曾煨杰.基于图像的铅笔自动生成算法研究与实现 [D].华南理工大学.2013:8-10
[3] Canny 边缘检测为什么用高斯平滑而他? blog.csdn.net/assjaa/article/details/103536420
[4] Canny 检测的 Matlab 实现(含代码)matlab 实现
blog.csdn.net/humanking7/article/details/46606791
[5] 冈萨雷斯. 数字图像处理（MATLAB 版）(第二版)

扫描以获取源代码文件及示例图像

感谢国家