南開大學
Nankai University

南　　开　　大　　学

计　算　机　学　院

计算机视觉基础实验报告

Canny Edge Detection

马琦

年级：2020 级

专业：计算机科学与技术

指导教师：程明明

2023 年 4 月 26 日

摘要

本次实验我主要实现了 Canny Edge Detection，选择的照片是经典的 Lena 图像。除了读取、显示图片等 I/O 操作其余功能均为手写。并且最后通过 trackbar 功能，测试了多组参数策略的效果。

实验的步骤主要分为四步，首先用高斯核过滤掉图片中的噪音，之后，再用 Sobel 算子求出图片在横向和纵向的近似梯度，得到图片的粗略轮廓图。这两步 Blur 的操作都可以通过 FFT 进行加速。此时图片的边界不够清晰，所以我们再用非最大值抑制将图片中不是极大值点的位置置为 0，其中对于梯度方向不在. 之后，我们利用双阈值 + 连边的思路，将图片中主要的轮廓留下来，将一些非主要的边界删去，这里连边主要是用泛洪算法实现。最终即可得到 Canny Edge，效果较好。

关键字：Canny Edge Detection、GaussianBlur、SobelBlur、FFT、Flood Fill

目录

一、 概述

(一) 实验原理

1. 滤波

Canny Edge Detection [1] 是图像边缘检测的重要算法，和之前用一个算子与图像做卷积不同，在本次实验中，我们先使用 GaussianBlur 和 Sobel 实现对图像的去噪和求梯度。其中 GaussianBlur 是典型的低通滤波器，其中的值均通过高斯函数求出，高斯函数定义如下:

$$f(x,y)=\frac{1}{2\pi\sigma_1\sigma_2}e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}-\frac{(y-\mu_2)^2}{2\sigma_2^2}}$$

而针对 Sobel 算子，则依赖于梯度的定义，Sobel 的思路在于分别求解横向和纵向的梯度，最后用三角形法则将两方向的梯度合成即可，对于 x 方向的梯度，用如下卷积核

$$\begin{bmatrix}-1&0&1\\-2&0&2\\-1&0&1\end{bmatrix}$$

对于 y 方向的梯度，用如下卷积核

$$\begin{bmatrix}-1&-2&-1\\0&0&0\\1&2&1\end{bmatrix}$$

其中为 2 的数值主要也是为了抑制噪声，所以加强当前位置处梯度比重。

这里手动实现了快速傅里叶变换，假设图像的边都是 N，卷积核大小是 M，(一般 M<N) 使得图像处理的理论复杂度降到 $O(N^2*log(N))$，与原来的 $O(N^2*M)$ 确实有所提升，其主要用到的是复数域内的周期性和关于原点的对称性，使得计算的次数显著下降，其主要原理如下:

设 ω_n^k 代表角度为 $\frac{2\pi}{n}k$，A 表示一个序列多项式，A_1 表示其中的偶数项，A_2 表示其中的技术项，则有

$$A(\omega_n^k)=A_1(\omega_{\frac{n}{2}}^k)+\omega_n^kA_2(\omega_{\frac{n}{2}}^k) \tag{1}$$

$$A(\omega_n^{k+\frac{n}{2}})=A_1(\omega_{\frac{n}{2}}^k)-\omega_n^kA_2(\omega_{\frac{n}{2}}^k) \tag{2}$$

这个其实就是分治 (也可以从归并的角度来看)，我在本次实验为了避免使用函数的递归，选择从低向上的循环算法 (也就是蝶形算法)，其原理图如下，

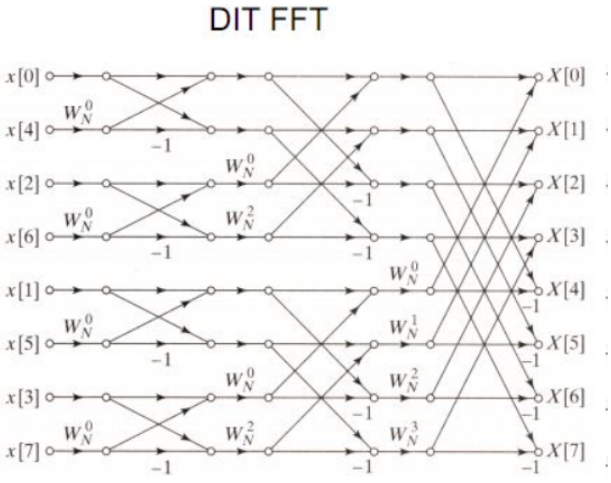

图 1: 蝶形算法的思路

将图像变化到频域域后，我们得到的是一个复数矩阵，同样的，我们将卷积核也变换过去(二者变换后的频域图应保持大小相同)，二者对位相乘后，再逆变换回来就可以得到滤波后的结果图。

2. 非最大值抑制

当我们进行上面的操作之后，会发现图像的边缘并不算清晰，会有很多个点均被识别为边缘，所以我们需要一种手段将边缘进一步细化，不再对相对不显著的边进行响应，这里就用到了非极大值抑制，我们对处理后的图进行分析，如果该点方向上的其他点的梯度均小于该点的梯度，那么我们就保留该点，否则我们抑制该点，从而实现边缘的精确检测。

当然，由于我们的图像并非连续的数值数据，有些点的方向上可能并不存在整数位置的数据，这时候，我们利用双线性插值，利用该点周围的四个点进行插值得到该方向上此点相邻的两个点，如果该点比这个相邻的点都大，那么我们选择保留该点，否则就抑制。

其中双线性插值的计算方法如下

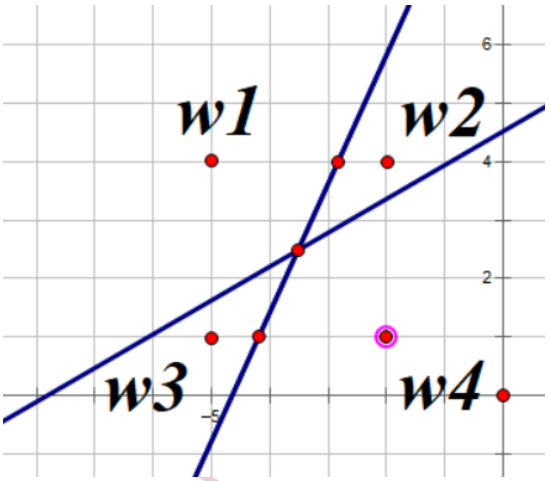

图 2: 计算原理

由上图所示，我们可以得到简单的一组关系，以直线交点落在上边的框为例，即该点梯度方向 (假设与 x 轴的夹角为 θ) 上的两个点的值分别为

$$val_1 = \frac{w_1 + w_2}{2} + \frac{w_2 - w_1}{2 * tan\theta}$$

$$val_2 = \frac{w_3 + w_4}{2} + \frac{w_3 - w_4}{2 * tan\theta}$$

通过这种方法，我们可以在离散的图像上获取近似的点周围的取值，进而更准确地判断该点是否需要被抑制，此方法天然带有该点的梯度信息，非常得有效。

3. 双阈值 + 连边

此处我们需要先设定图像的两个阈值，一个阈值用来确定连边的源点，另一个阈值用来确定需要被过滤的点，两个阈值之间的值用来辅助判断是否可以被源点所传播，即只有在图上和源点处于同一连通分支内 (源点必须是高于高阈值的点，且一个分支内必须存在源点) 才可以连接，此方法可以有效地将一些中间干扰点 (即强度在高阈值和低阈值之间但是非边界的点) 滤去。

这里我们使用 FloodFill 算法来实现对图片中主要边界点的连接，我们将高于阈值的点当做源点，从源点开始向四周 (即通过 BFS) 进行泛洪传播，同时进行记忆化标记，这样我们可以在最短的时间内实现所有连通分支的连接。

(二) 实验素材

本次实验主要选择 Lena 图作为实验素材，原因在于这张图片最复杂，可以充分体现我们算法的实现效果。

图 3: 图像处理的耶路撒冷: Lena 图

(三) 实验环境

- 操作系统：Windows10
- OpenCV 版本：4.6.0
- IDE:Visual Studio 2022

二、 实验结果

(一) 结果呈现

为了排除颜色信息的干扰，我们使用灰度图作为处理对象

图 4: Lenna 的灰度图

对于给定的双阈值，我们的 Canny 算法实现的效果如下：

图 5: 最大值抑制后的结果

图 6: 连边后的结果

图 7: 连边的效果图

图 8: OpenCV 库自带的 Canny 函数

非常难得的是，我的效果比官方 OpenCV 的效果略微好一些，可能是运气吧，OpenCV 库

的 Canny 可以发现有一些细碎的信息，同时丢失了左边的柱子，感觉是对图像的预处理有问题，也有可能是我阈值设置得不好。

与此同时，我也实现了 Ostu 全局阈值 [2] 和 Sauvola 局部阈值 [3]，并且对二者进行了结合，结果如下：

图 9: 单纯全局找阈值的效果

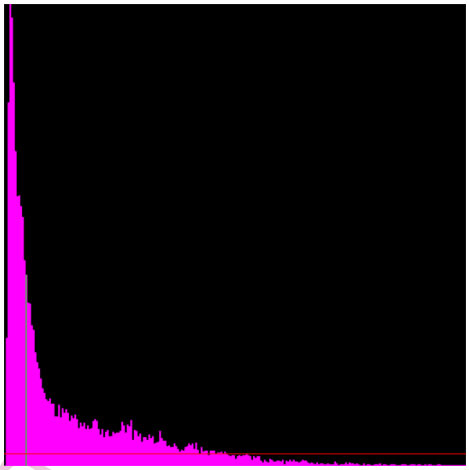

图 10: 全局找到的阈值对应的灰度值

图 11: 全局结合局部后进行二值化

图 12: 结合后的效果

(二) 结果分析

1. 阈值的选取

双阈值选取的要点主要在于，对于高阈值，不应该太高，那样会丢失大部分源点，连出来的边界是残缺的，同时也不宜太低，那样会使得噪音太多，边界不明显。对于低阈值来说，主要考虑的事能否将无用的孤立点信息删去，同时能否将有用但是比较微弱的边信息保留，这些都需要通过具体的图片进行分析。

2. 自适应阈值的求解方法

起初我只是分别实现了两种自适应算法，其中，局部自适应非常得差，由于阈值改为了局部几个正方形区域内的均值和方差，会受到黑色区域的极大干扰，比如这个区域内全是黑的，那么阈值就是 0，这样的话所有的点都会被标记，但是我们又不能完全不考虑黑色的区域，正是因为

纯黑区域的存在才能使得边界信息愈发突出，考虑到实现的 Ostu 算法能够很好地找到全局中有用和无用信息的分界线，我尝试将 Ostu 求解的阈值作为过滤器，将局部自适应得到的区域信息进行进一步的处理，最终得到的效果如图 12 所示，既滤去了太多无用的信息，同时又保留了一些局部的有用边界，同时还不需要手动进行调参，一举多得。

三、 代码链接

手动调参：DEBUG-Canny.cpp
自适应阈值：self-Adaptive-Canny.cpp

参考文献

[1] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[2] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[3] Jaakko Sauvola and Matti Pietikäinen. Adaptive document image binarization. *Pattern recognition*, 33(2):225–236, 2000.