数字图像处理第三次作业实验报告

181860152 周宇翔

实验内容

边缘检测:

本次实验,我一共使用了三种方法进行边缘检测,分别是使用sobel算子进行基本边缘检测,Marr-Hildreth边缘检测算法,以及Canny边缘检测算法.其中Marr-Hildreth算法可能由于个人实现的问题在给定的这组图片集上表现不佳,故在此只对另外两种方法进行分析

Sobel算子基本边缘检测 (BasicEdgeDetection.m)

基本边缘检测的思想很简单,只需要用sobel算子对图片进行卷积即可,然后用x和y方向的梯度计算出图片每个点的梯度和方向即可.由于需要生成二值图像,需要自己选取阈值,阈值的选取是比较难的,因为对每张图来说自动生成的阈值并不是很普适,比如对于条纹密集型图片(比如leaf.jpg giraffe.jpg)选取的阈值应该相对偏高,而对于有明显的大边缘的图片(比如婆崎的那两张图)选取的阈值应该相对偏低,我并没有找到很好调和这种矛盾的方法,因此只能手动设置阈值(狗头保命)

Canny 算法(Canny.m)

Canny的实现比较复杂,先用高斯滤波器滤波后再用sobelx,sobely计算图像每个点两个方向的梯度,进而计算出每个点的梯度和法向,通过非最大抑制确定边缘点和准边缘点,最后通过8连接最终确定边缘.

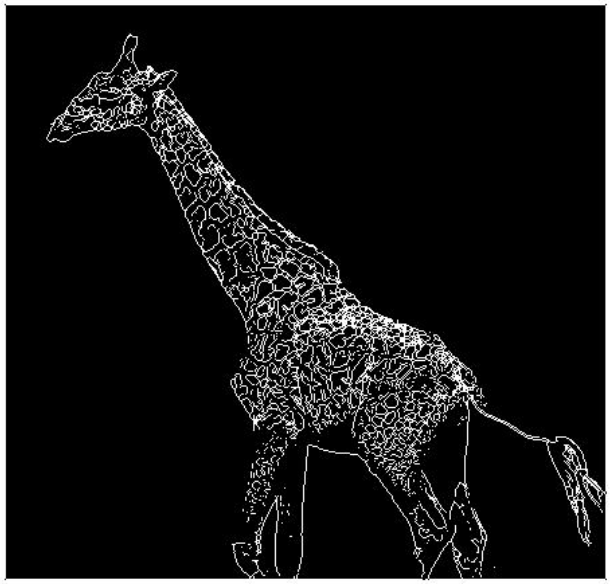
其中的高斯滤波器我仍采用的是此前自己实现的高斯滤波器,故性能可能没有Matlab自带的滤波器性能好.最大的问题任然是在于阈值的选择,在非最大抑制之后的阈值处理部分,我们需要选择低阈值T_L和高阈值T_H

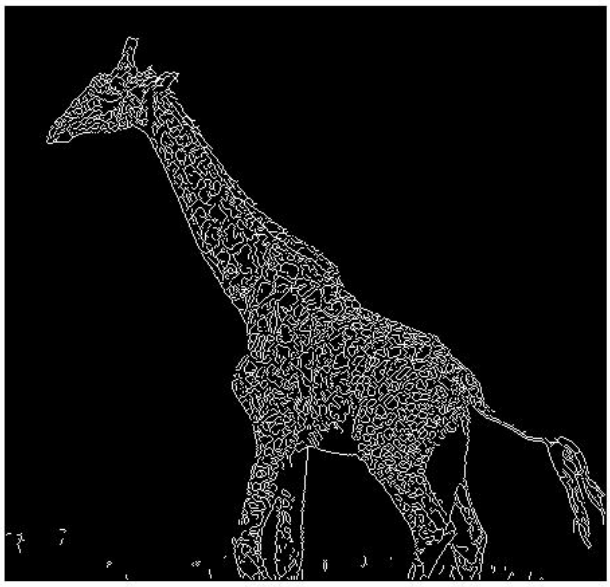
为了在所有的图片上都取得一个比较好的效果,我进行了一些类似于调参的过程,最后选取阈值如下

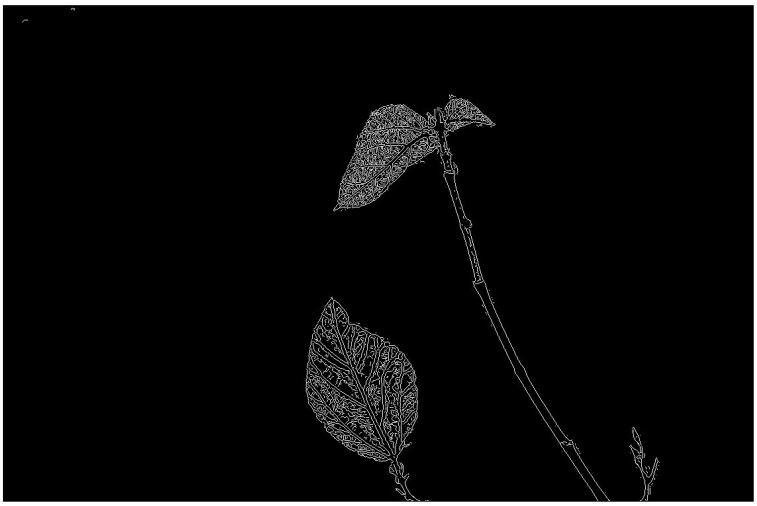
```
TL = sum(gN(:))/sum(sum(gN~=0));
TH = 2.25*TL;
```

也即低阈值为所有非零像素的平均,而高阈值为低阈值的2.25倍,也符合我们通常的高阈值/低阈值在2-3之间的假设

所有图片的处理结果如下,左为BasicEdgeDetection,右为Canny

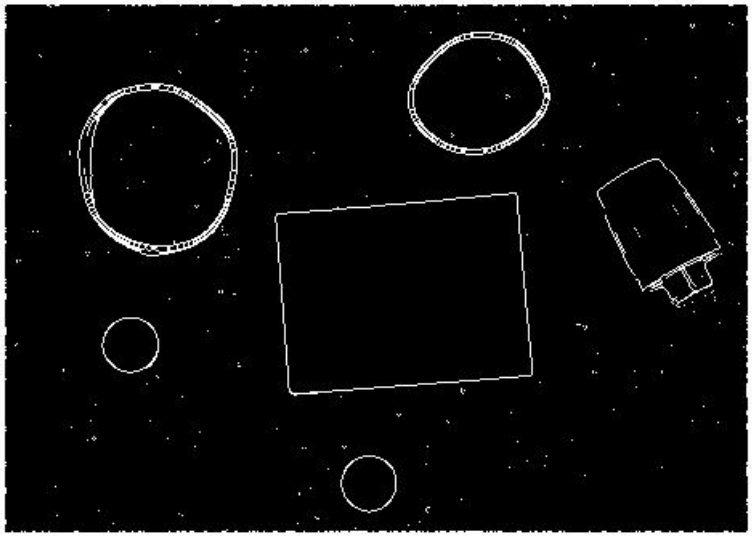


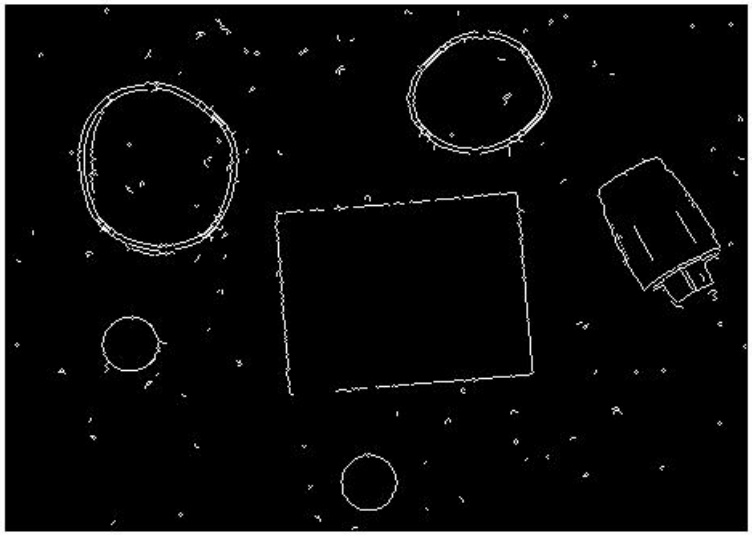





AYUMI HAMASAKI
BEST
A
B


AYUMI HAMASAKI
BEST
A
B





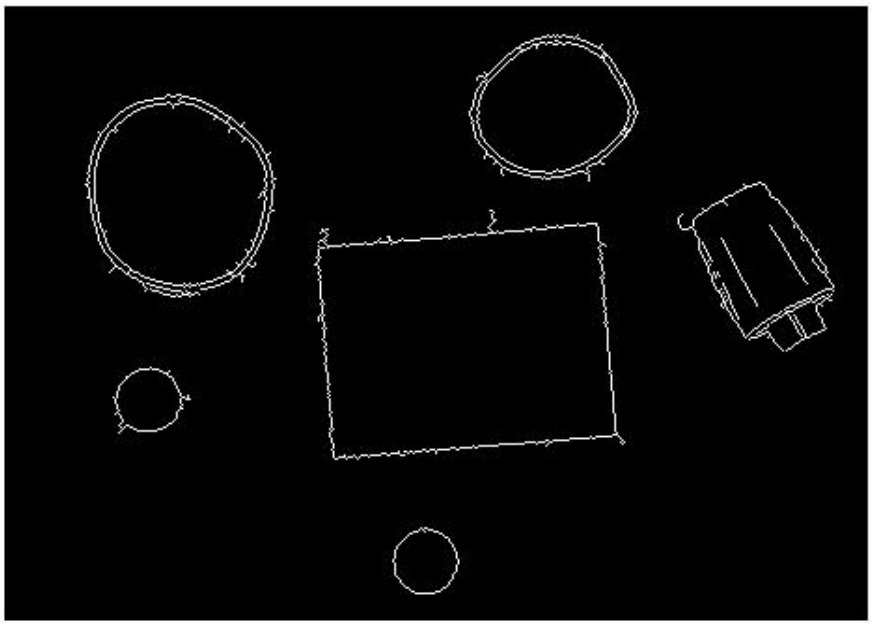
可以看到的是,利用BasicEdgeDetection生成的边缘往往包括一些粗线条,而Canny的线条基本上都是一个像素点构成的细线.Canny能够包括更完整的边缘信息,但同样也会引入更多的噪声,以及细碎的边缘.不过这些都可以通过阈值的调整来取得更好的效果,因此我认为设置更好的阈值选取算法是比较重要的

边缘连接:

在此次实验中只要求对rubberband_cap.png进行边缘连接,我采用的方法是摩尔邻域算法,它的大致思路是

给定图片中某个对象的边缘上的一个点,通过对它的八邻域扫描来确定下一个边缘上的点,同时记录下我们这个推进的方向是哪个方向,直到到达起点且推进的方向和初始推进方向一致为止.

这个方法在本图中比较好用的原因,是因为图中的边缘大多是简单的几何图形且相对连续,我们通过降低阈值,先用canny生成边缘较连续但噪声也较多的边缘图,由于我们可以用imtool来手动定位边缘,这些噪声最终都可以忽略不管,最终我们生成如下的边缘连接图



摩尔邻域算法的优点在于实现相对简单,并且计算开销小.但是它的缺点也相当明显,它的robustness比较差,很多时候它很容易就陷入死循环,在一小圈像素内不停来回;并且对于中断的边缘(哪怕只有一个像素)也无法处理,我试过让它能够在检测到一个像素中断时去自动连接,但这样会引入更多的死循环问题,因此只能通过增加手动定位的边缘点,让该算法多次进行来完善边缘图像.因此尽管它在这个图像上还算实用,但在更广泛的应用场景下的表现还有待考证.