总体思路：

首先要得到A4纸的边缘，采用canny边缘检测算法，得到边缘，要注意的是对sigma和threshold两个参数的取值。

接着要得到边缘的直线方程。在第一步的基础上用霍夫变换，得到每条边缘的直线方程。要注意的是对点的投票值的取值。

要得到每个点的坐标，我的做法是类似于投票法，当然也可以利用第二步得到的直线方程，但是这样要考虑的东西比较多，所以偷了个小懒，不过图片经过下采样之后像素点并不是很多，效率也很高。

各个参数意义：

in\_thresh主要用于去除噪声，低于像素值低于此的不进行霍夫变换

out\_thresh取值区间为0~1，在霍夫空间中与最大点的比值低于此的不认为是一条直线

theta\_difference两条直线的theta值差距在此之间认为两条直线可能相等

offset\_difference两条直线的offset值差距在此之间认为两条直线可能相等

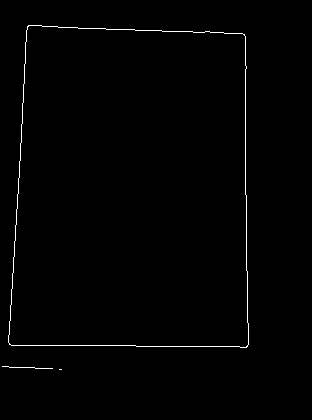
non\_maximum\_suppression对霍夫空间进行取点的时候，对某个点的周围的几个点组成的点阵进行判断，取出点阵中值最大的点，避免直线重复

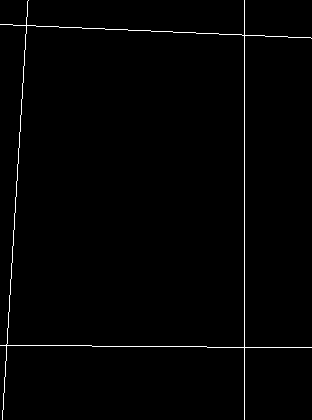
思考如何运行得更快：

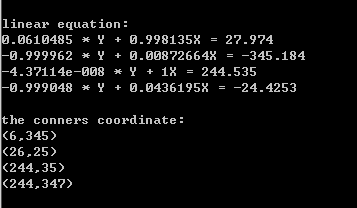
1. 将图片进行适当的下采样，可有效降低运行时间
2. 确定合理的梯度步长
3. 计算点的位置的时候不用投票法，直接进行计算

第一张图片的运行结果：

参数取值：sigma=2.0，threshold=10.0，in\_thresh=223.0，out\_thresh=0.50 ,theta\_difference=0.2,，offset\_difference =5, non\_maximum\_suppression=0

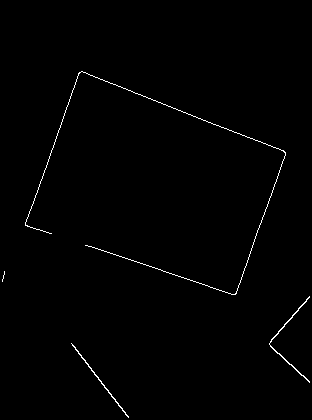
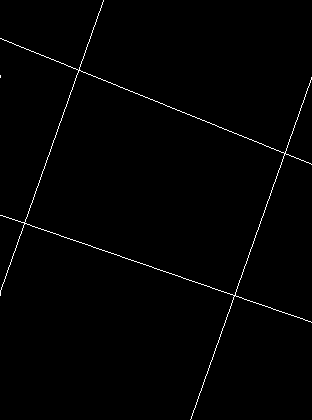


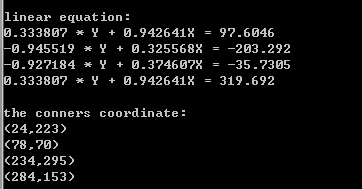




第二张图片：

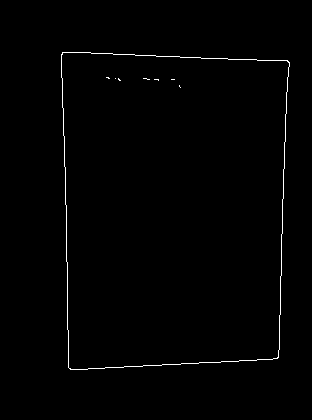
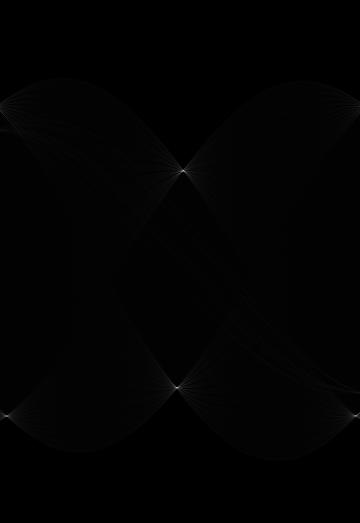
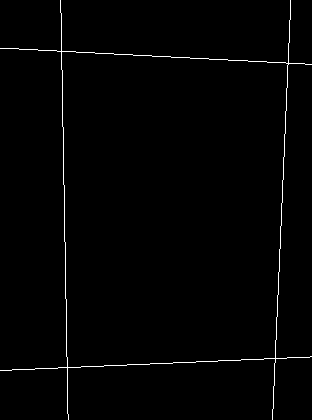
参数取值：sigma=2.0，threshold=15.0，in\_thresh=223.0，out\_thresh=0.75 ,theta\_difference=0.2,，offset\_difference =5 ,non\_maximum\_suppression=0

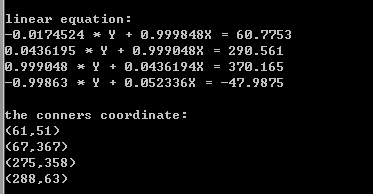
 



第三张图片：

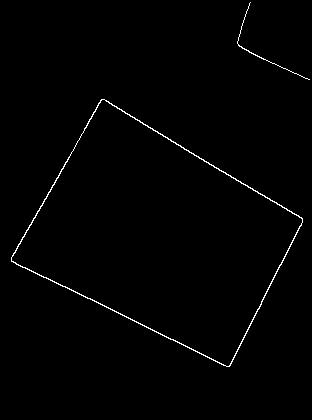
参数取值：sigma=2.0，threshold=10.0，in\_thresh=223.0，out\_thresh=0.50 ,theta\_difference=0.2,，offset\_difference =5 ,non\_maximum\_suppression=0

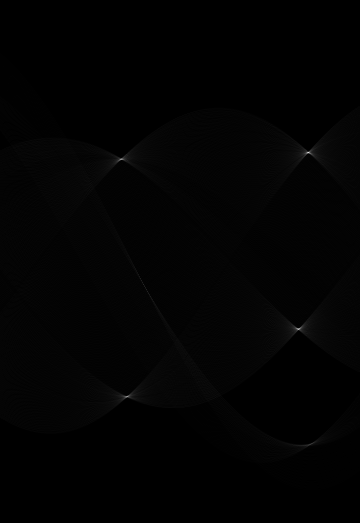
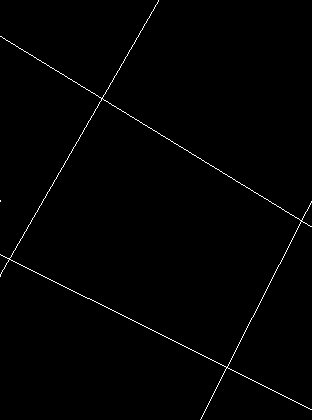
  

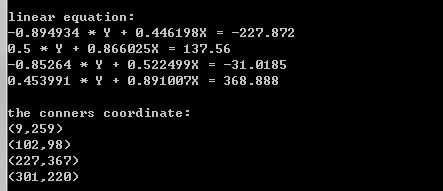


第四张图片：

参数取值：sigma=2.0，threshold=10.0，in\_thresh=223.0，out\_thresh=0.50 ,theta\_difference=0.2,，offset\_difference =5 ,non\_maximum\_suppression=0

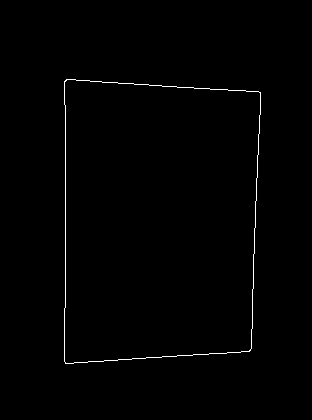


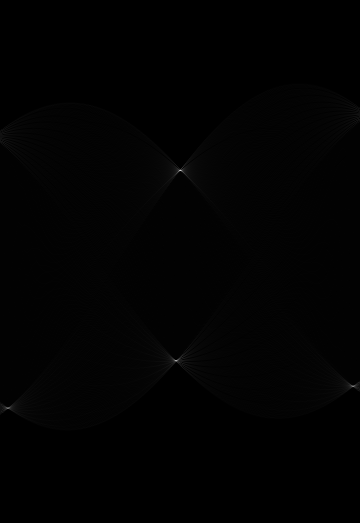
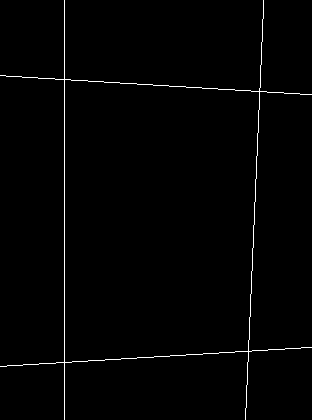
 

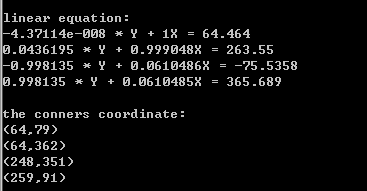


第五张图片：

参数取值：sigma=2.0，threshold=15.0，in\_thresh=223.0，out\_thresh=0.75，theta\_difference=0.1,，offset\_difference =3, non\_maximum\_suppression=1

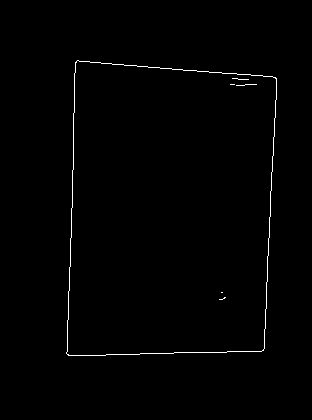
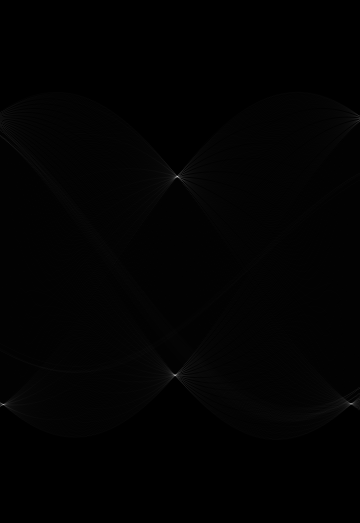
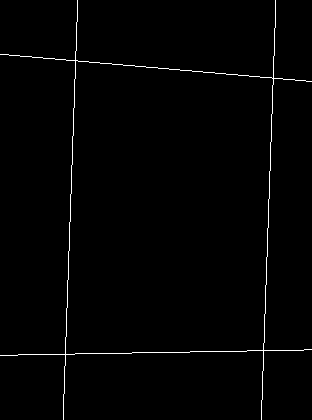
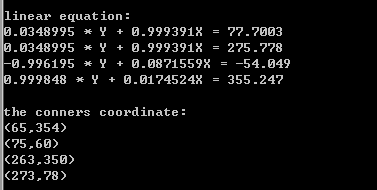




第六张图片：

参数取值：sigma=2.0，threshold=10.0，in\_thresh=223.0，out\_thresh=0.50, theta\_difference=0.2,，offset\_difference =5, non\_maximum\_suppression=0

总结：这次的作业很多坑，我做了很久很久，原因如下：

1. 一开始没有将图片进行下采样，导致每次跑出结果要等很久很久。
2. 要对每张图片的每个处理过程的每个参数进行调教，特别痛苦。
3. 我是在Windows下做的作业，没有makefile的参与，导致每次修改参数后进行编译要等一分钟