

《计算机视觉——算法与应用》课程

大作业

Hough变换直线检测算法实现

题目：

学 号：

刘锦维

1221045808

姓 名：

计算机学院

学 院：

电子信息

专 业：

肖学中

任课教师：

日期： 2021年12月27日

## Hough变换直线检测算法实现

**摘 要：**霍夫变化（Hough Transform）自1972年推广以来，已成为图像处理领域内，检测图像中的几何形状的基本方法之一。Hough直线检测的基本原理在于利用点与线的对偶性，即图像空间中的直线与参数空间中的点是一一对应的，参数空间中的直线与图像空间中的点也是一一对应的。依次遍历图像上的直线，确定直线上点的是数目，选择出合适的直线，就可完成直线检测。本文使用opencv，完成相关的Hough变换直线检测的代码实现，并实际测试该算法的相关性能。

**关键词：**Hough变换；直线检测；opencv。

# 1 概述

霍夫变换于1962你说呢由Paul Hough首次提出，后于1972年由Richard Duda和Peter Hart进一步研究与推广，现如今，已成为图像处理领域当中，用于图像内几何图形检测的基本方法之一。

Hough变换直线检测是一种检测图像上有无直线的一种基本方法该方法是一种使用表决方式的参数估计技术，其原理是利用图像空间和Hough参数空间点与线的对偶性，即图像空间中的直线与参数空间中的点是一一对应的，参数空间中的直线与图像空间中的点也是一一对应的，把图像空间中的检测问题转换到参数空间中进行。通过在参数空间中找到峰值，或使用一些约束条件来完成直线检测任务。

本文重点在于阐述Hough变换直线检测的基本原理和算法步骤，并使用python及相关的模块来实现该算法，并检测该算法在实际图片检测中的性能表现。

# 2 算法原理

Hough变换相比其他当今的图像直线检测算法，虽然原理相对简单，却具有十分重要的启蒙作用。

# 2-1 对偶性

1. 图像空间中的点与参数空间中的直线一一对应

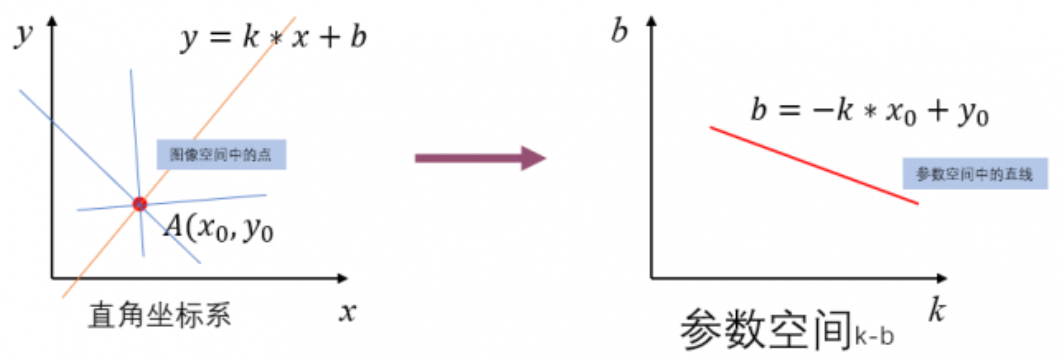
在图像空间X-Y中，任意一个像素点都可以在直角坐标系下表示为,同理，一条直线则可表示为：

其中k和b是参数，表示直线的斜率和截距。而过某一点的所有直线的参数均应当满足方程，即点可以确定一个直线簇。

如果我们将方程改写为：

那么该方程在参数空间*k - b*中就对应了一条直线。也就是说图像空间*x - y*中的点对应了参数空间*k - b*中的直线 。因此可以得出结论，图像空间中的点与参数空间中的直线一一对应。

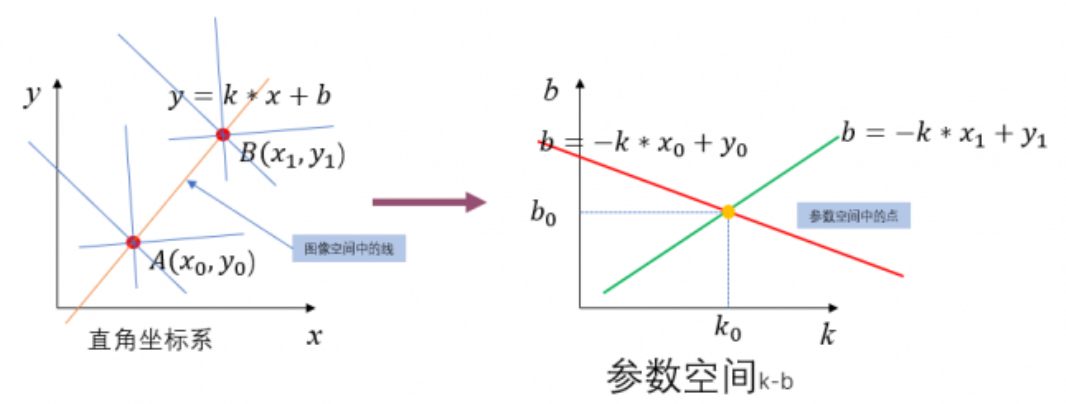
上诉过程可以表示为下图：



图表 1 图像空间映射到参数空间

1. 图像空间中的直线与参数空间中的点一一对应

在图像空间*x - y*中，我们再在直线上增加一个点,即可确定某一条确定的直线，同时点在参数空间*k – b*上同样会对应一条直线，如图二所示：



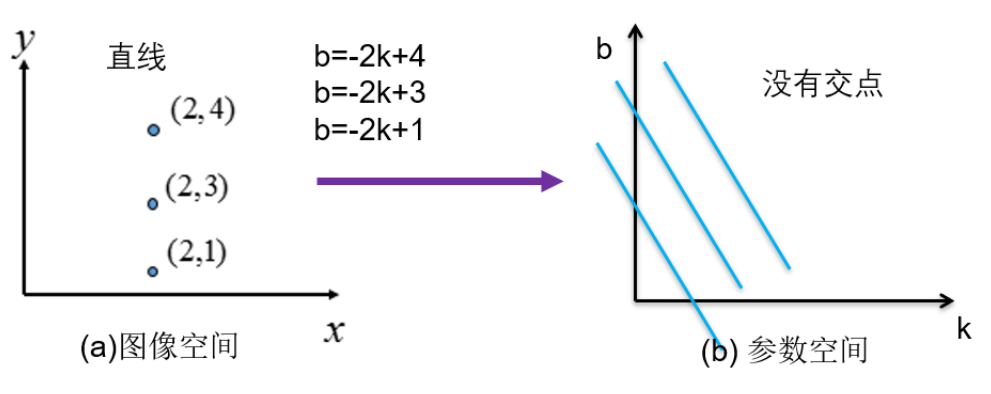
图表 2 加入点B

通过观察上图不难看出，图像空间*x – y*中的点A和点B在参数空间*k – b*上对应的直线相交于一点，即AB两点确定的一条直线，在参数空间中将对应唯一一个点，而该点的坐标值也就是直线AB的参数。

而这个对偶性质为解决直线检测任务提供了方法，即把图像空间中的直线对应到参数空间上，最后利用统计特性来解决问题。假如图像空间中有两条直线，那么最终在参数空间中就会对应到两个峰值点，依此类推。因此可以得出结论，在参数空间上，相交直线最多的点，对应图像空间*x – y*上一条确定的直线。

# 2-2 参数空间的选择

上述为讲解图像空间和Hough参数空间点与线的对偶性这一重要原理，参数空间采用的是直角坐标系。但在实际应用中，参数空间是不可选择直角坐标系的。即原始图像直角坐标空间中存在特殊的直线，该直线垂直于X轴，直线的斜率为无穷大，而这是无法在基于直角坐标系的参数空间中表示的。即这条直线上的点映射到参数空间上的直线为平行线，彼此之间不存在任何交点。具体情况可表示下图：



图表 3 直线x = c

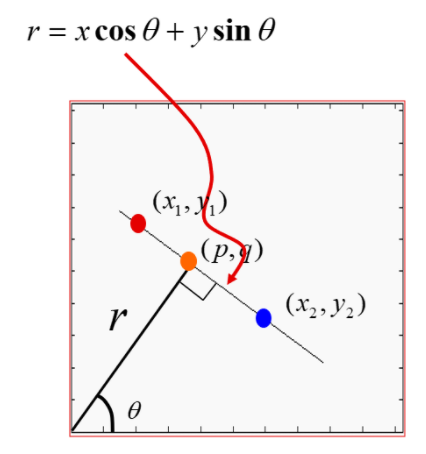
为解决上述问题，参数空间一般采用极坐标方式表示。具体转换过程如下：

原有图像空间上的任意一点在极坐标系下表示为：

任意一条直线的垂线与X轴的夹角为，而该直线的与X轴的夹角为,则其斜率为：

设直线上的任意一点为，则斜率为：

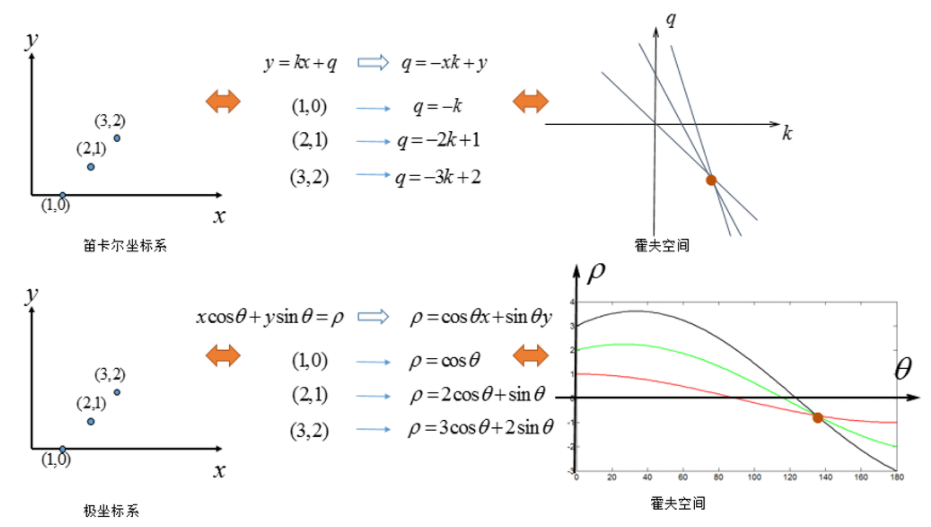
根据（4）、（5）两式，可以得到：



图表 4 坐标系转换

从（6）式可以看出，参数空间中的每个点都对应着图像空间中的一条直线，而这也有效避免了使用直角坐标系无法表示一些特殊直线的尴尬情况。但值得关注的是，原有图像空间中的点在参数空间中，均表示为一条正弦曲线，两条曲线的交点表示图像空间两点确定的一条直线。

具体实例可以参考下图：



图表 5 两坐标系映射比较

# 2-3 检测直线

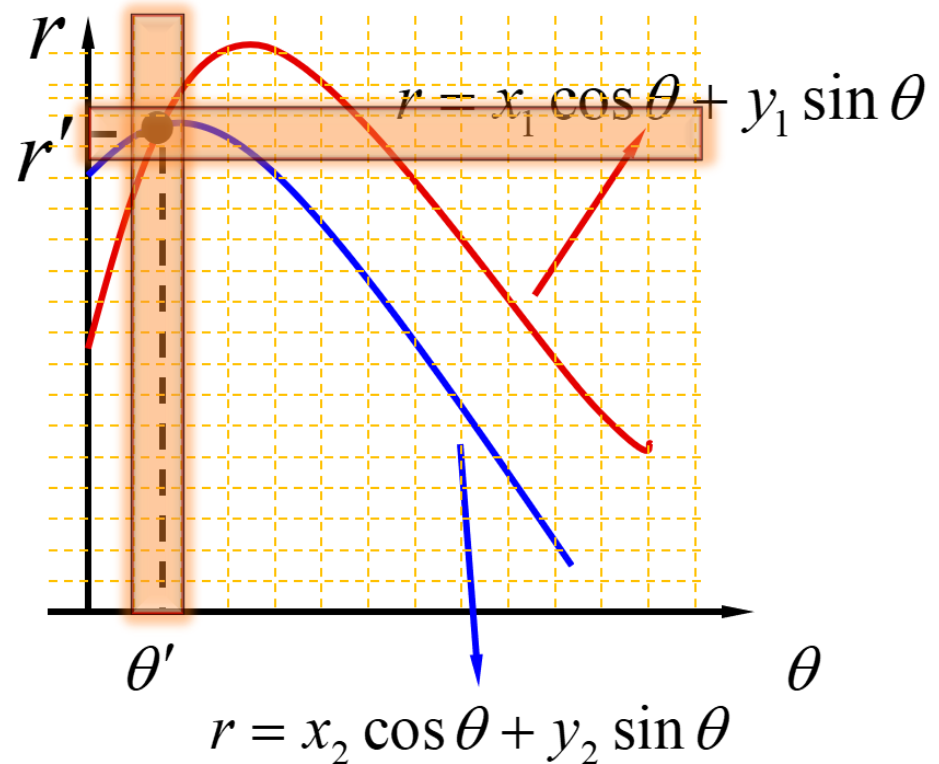
采用霍夫变换把图像空间中存在的直线变换为参数空间中的点，利用统计特性便可用于解决直线检测任务。具体来说，即原图像空间中的像素点映射到参数空间中，变为一条曲线，而曲线之间的交点这表示有两个过多个像素点确定的有一条直线。

而由上述说明，可以确定原有图像还不能直接用于霍夫变换直线检测，还需进行去噪、灰度处理和边缘检测等操作。

同时，需要注意的是，在理论上，图像空间上的一个点对应着无数的直线（在参数空间上，则是一条曲线上有着无穷个点），而这会造成计算量过大，程序陷入死循环的困境当中。因此，有必要对参数空间的两个坐标和进行离散化，是两者的取值有限个，限制直线的选取数量，才可以完成相关检测任务。

参数空间坐标参数的离散化，将使得参数空间离散化，被分割为一个个等大小的网格单元。将图像空间中的像素点变化到参数空间中，会经过一定数目的网格，并将这些网格的累加计数器加1 。当图像空间中的所有的像素都经过霍夫变换，对网格单元进行统计，取累加计数值最大的网格，其坐标就对应这图像空间中的某一条确定的直线。

离散化后的参数空间可由下图表示：



图表 6 离散化显示、

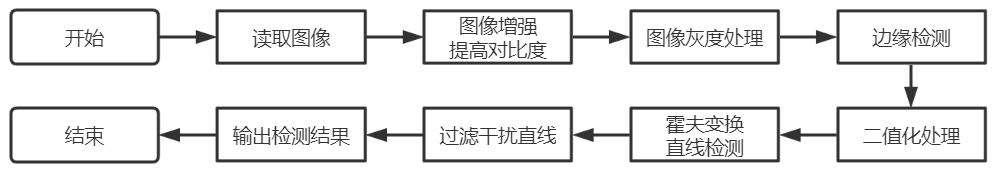
# 3 算法实现

# 3-1 算法流程

根据霍夫变换直线检测的算法原理，先要提取出原有图像上的像素点，而噪音和过于杂乱的像素点会对该算法最终的检测结果带来巨大的负面影响。为解决这些问题，一般需要原始的彩色图像进行灰度处理，使该图像由RGB图像变换为灰度图像，再对该图像去噪和边缘提取，最后对处理过的图像进行二值化处理，以灰度值为255的边缘点进行霍夫变换。

之后在统计特性，设定阈值，选取合适的直线参数。

本文具体步骤，如下述的流程图所示：



图表 7 算法流程

# 3-2 算法实现

Python作为有一种常用的程序设计语言，集成了大量模块库，可以对编程带来许多便利。其中Opencv模块中，包含了大量的图像处理函数，其中就包含了像图像灰度处理、图像二值化处理、图像边缘检测及Hough变换直线检测等函数，可以快捷且简便地完成相关检测任务。

在该算法中，首先需要将RGB图像转变为灰度图像，可以直接调用Opencv中的cvtColor函数，该函数以待检测的图像作为输入，可根据第二个输入将待检测图像转化为对应的图像形式，其中cv2. COLOR\_BGR2GRAY代表灰度图像。

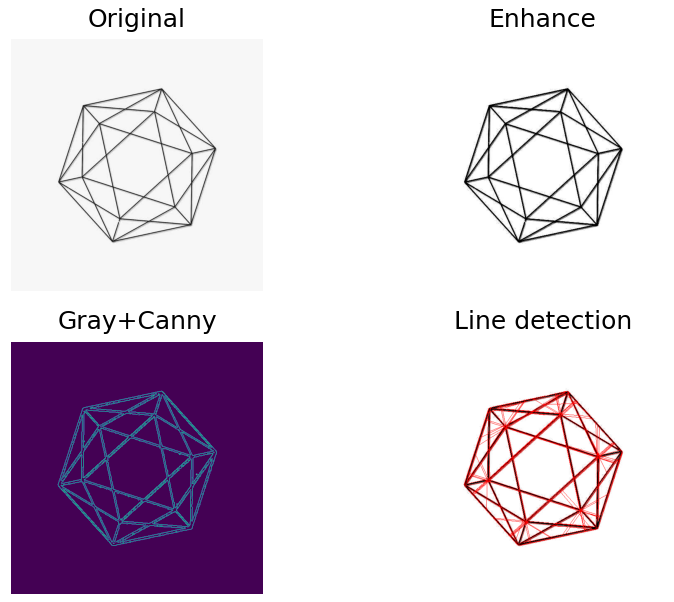
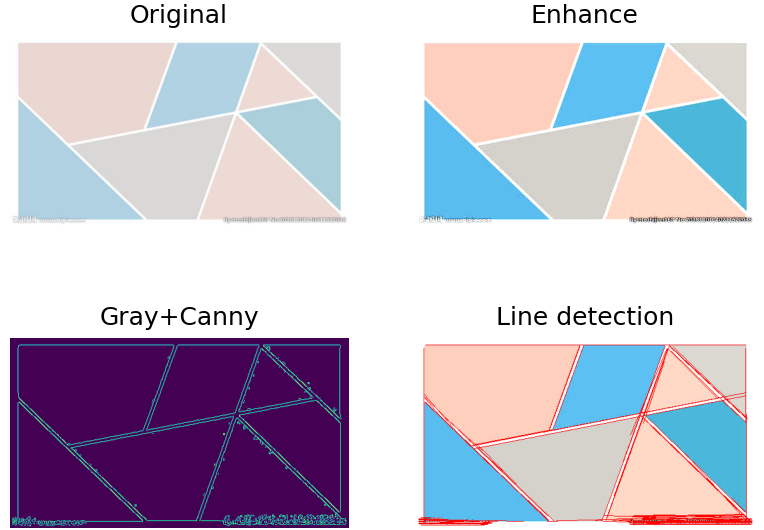
而threshold函数则是Opencv常用的二值化处理方法，该方法可以依据某一阈值，将图像进行二值化处理 。而该步骤是为了方便图像的边缘检测。同时Opencv库中提供了Canny函数可以用于边缘检测，该函数使用sobel算子，计算出每个的点的梯度大小和梯度方向，并使用非极大值抑制，消除边缘检测带来的杂散效应，再应用双阈值，来确定图像中真实和潜在的边缘，最后通过抑制弱边缘，即可完成边缘检测任务。

Opencv库中提供了两种方法可用于Hough变换直线检测，本文是使用了HoughLinesP函数完成该步骤。该方法含有6个参数，第一个参数表示经过灰度处理及边缘检测后的二值图像，而第二个和第三个参数则表示r和的精度，第四个参数表示Hough变换累加器的阈值，高于该阈值的直线才被保留，而最后的两个参数表示线段最小长度及线段间的最大的间隔。

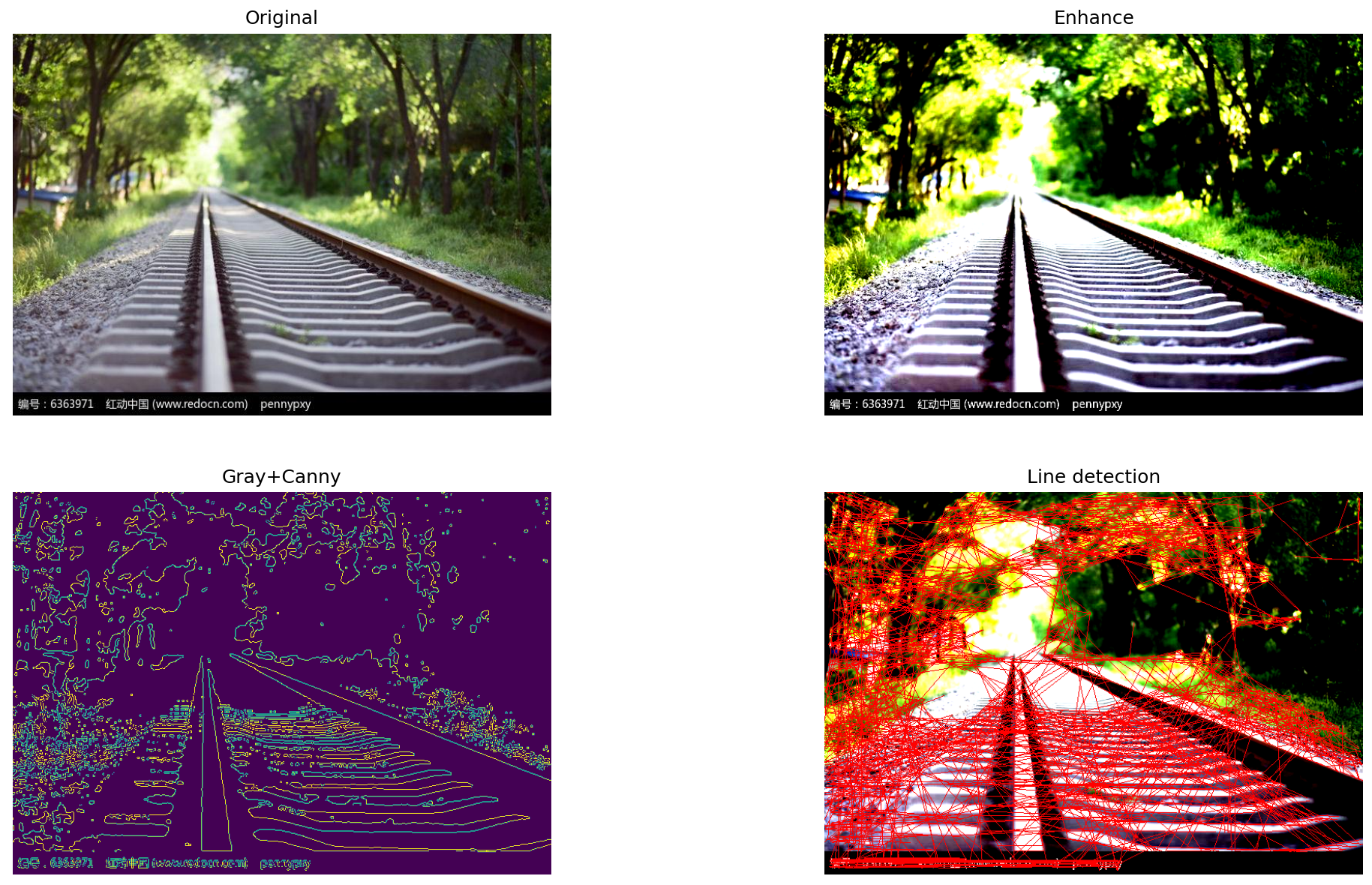
最后再标注出这些检测出来的线段，即可完成直线检测任务。

# 4 算法结果

调用指定的程序，对一些预先设定的图片进行霍夫变换直线检测，检测结果如下图所示：



（a） （b）



（c）

图表 8 Hough变换直线检测结果

由以上检测结果，可以发现Hough变换直线检测可以对较为简单的图片和几何形状表现出一个较为优异的检测效果。但对于一些复杂的图像，该算法无法过滤掉绝大部分的干扰直线；而对于带有模糊的图像，根据人的主观认知及经验，一些重要的直线会在边缘检测中，丢失掉大量的相关信息，使得后续的检测中，会被忽略；而且参数空间的离散化间距也会制约该类型算法的检测精度。其次由于Hough变换的特点，该类型算法的时间复杂度和空间复杂度都很高，计算开销较大。

上述的几种问题都是Hough变换直线检测中难以处理情况，还需进一步改进。

# 5 结束语

Hough变换直线检测是一种传统且简单的一类算法，利用了图像空间和参数空间中点与线的对偶关系，并利用统计特性可以很快的得出待检测图像的直线检测结果，但是该类算法只对一些较为简单的几何图形和一些环境较为简单的图像可以表现出优异的效果。但是，该算法较为依赖边缘检测结果，因此对于一些较为复杂的图像，难以取得令人满意的结果。

如要想获得较好的表现效果，还需对图像进行合理的预处理，或结合一些其他方法。