边(Features)

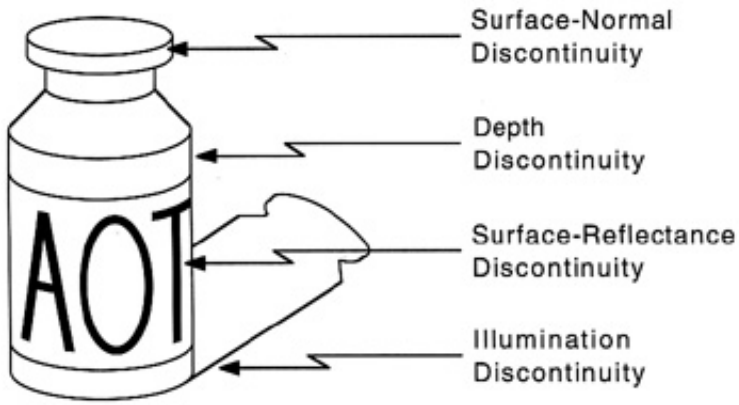
边缘线条会给我们带来大量关于**图像信息**，线条可以事物之间**边界线**，也可以事物阴影区域都是线条。边缘检测用于图像分割，提取图像形状的信息，那么边缘检测也就是用二值图表示出图片中边缘信息。所以在图像中，线条相对于颜色和纹理，显得格外重要。能够给我们带来大量的信息。我们无需色彩，仅单色的线条图就可以清楚来表达一些信息，而且这些信息足够反应图像的语义。线条位置也就是亮度变化明显的位置，或者颜色变化位置，下面介绍几种变化的模式共大家参考。

- 边缘更紧凑
- 边缘属于对象级别的信息

虽有通过无人驾驶中识别车道线例子来介绍边缘检测应用，其中会用到 Canny 边缘检测和霍夫变换。

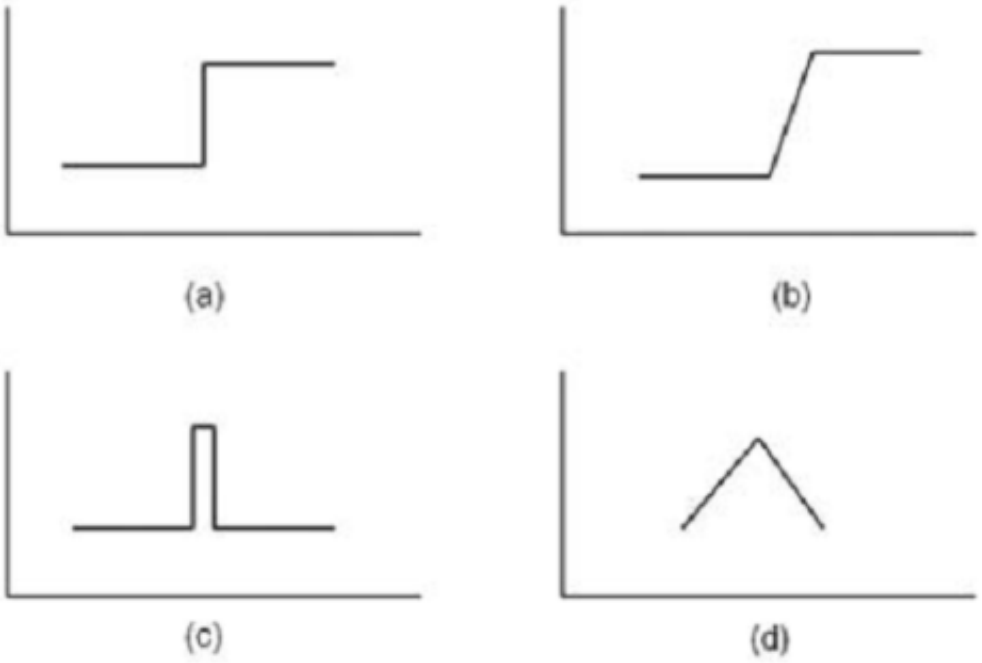
边缘产生的位置

这里来简单地介绍一下，我们如何判断是边缘，要判断什么是边缘，首先需要知道什么是边缘，为什么会有边缘的存在。边缘定义——图像强度的不连续性就是边缘，那么什么是强度呢，强度暂时可以理解为灰度。



- 表面法向不连续性(surface normal discontinuity)
- 景深不连续性(depth discontinuity)
- 表面颜色不连续性(surface color discontiuity)
- 光照不连续性(illumination discontinuity)

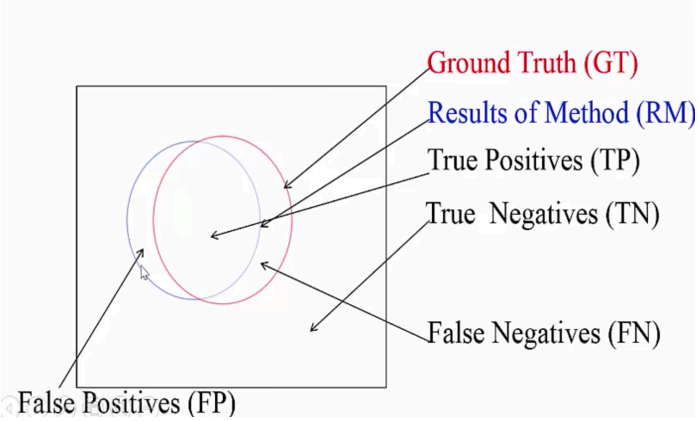
线类模型



- Step(a) 阶跃边缘模型，也就是变换一下从比较暗区域跳跃到明亮的区域，这是一种明暗变化的模式
- Ramp(b) 斜坡边缘模型
- Spike(c)
- Roof(d)这种变换相对于上面 Step 阶跃变化变化不会那么突然，变换相对平稳，先是平稳上升接下来是平稳下降的过程

线检测的标准

其实在 AI 项目中，比较重要的是先定义出一个标准，有了标准才能开展随后的工作。有了标准才能看出好坏，这里标准适合大多数的机器学习项目。对于分类任务和预测任务是比较通用方法也适合边缘检测的能力。



- Results of Method (RM) 这个区域表示算法识别出的
- Ground Truth (GT): 表示图像中实际为边缘的区域
- True Positives (TP): 表示模型将边缘正确识别的区域
- True Negatives (TN): 表示不是边缘的检测为不是边缘
- False Positivies (FP): 表示将不是边缘检测为边缘
- False Negatives (FN): 表示模型没有检测出的边缘区域

$$precision = \frac{GT \bigcap RM}{RM} = \frac{TP}{RM}$$

$$recall = \frac{GT \bigcap RM}{GT} = \frac{TP}{GT}$$

边缘检测分类

一阶导数的边缘算子

通过模板作为核与图像的每个像素点做卷积和运算，然后选取合适的阈值来提取图像的边缘。接下来介绍的 Roberts Cross、Sobel 算子和 Prewitt 算子都属于一阶导数的边缘算子。

二阶导数的边缘算子

二阶导数过零点，常见的有 Laplacian 算子，此类算子对噪声敏感。

其他边缘算子

前面两类均是通过微分算子来检测图像边缘，还有一种就是Canny算子，其是在满足一定约束条件下推导出来的边缘检测最优化算子。

边缘检测的算法

接下里我们会列出一些边缘检测的算法，例如 Roberts、Prewitt 和 Sobel 算子，还会介绍 Canny 算子的边缘检测。首先我们都知道明暗(或者说强度)变化大位置为边缘。那么如何检测一张图片强度变化，这里需要引入导数，但是又因为图像是离散的所以对于离散型函数进行求导还有些特别方法。这里我们一步一步地展开来来看如何进行对图像强度进行求导。

在一维连续数集上有函数f(x),我们可以通过求导获得该函数在任一点的斜率，根据导数的定义有

$$f'(x) = f(x + \Delta x) - f(x)$$

在二维连续数集上有函数f(x,y),我们也可以通过求导获得该函数在x和y分量的偏导数，根据定义有：

$$\frac{\partial f(x,y)}{\partial x} = f(x + \Delta x, y) - f(x,y)$$
$$\frac{\partial f(x,y)}{\partial y} = f(x, y + \Delta y) - f(x,y)$$

对于图像来说，是一个二维的离散型数集，通过推广二维连续型求函数偏导的方法，来求得图像的偏导数，即在(x,y)处的最大变化率，也就是这里的梯度

$$g_x = \frac{\partial f(x,y)}{\partial x} = f(x+1, y) - f(x,y)$$
$$g_y = \frac{\partial f(x,y)}{\partial y} = f(x+1, y) - f(x,y)$$

接下里我们就用梯度形式来

$$\nabla f = grad(f) = [g_x, g_y]^T$$

对于梯度是由方向和大小的，好那么什么又是梯度的大小、什么又是梯度的方向呢。接下来用公式给大家展示一下。

$$M(x,y) = mag(\nabla f) = \sqrt{g_x^2 + g_y^2}$$

为了减小开销可以将梯度模表示为

$$M(x,y) = |g_x| + |g_y|$$

方向表示为

$$\alpha(x,y) = \arctan\left[\frac{g_y}{g_x}\right]$$

- 向后差分 [-1,1]
- 向前差分 [1, -1]
- 中心差分[-1,0,1]

二阶求导

Roberts Cross

Roberts算子(Roberts Cross)又称为交叉微分算法，这是比较早的一个边缘检测的算子。是基于交叉差分的梯度算法，通过局部差分计算检测边缘线条。常用来处理具有陡峭的低噪声图像，当图像边缘接近于正45度或负45度时，该算法处理效果更理想。其缺点是对边缘的定位不太准确，提取的边缘线条较粗。

$$G_x = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} G_y = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Prewitt

对于 Prewitt 边缘检测，首先进行平滑处理

$$image \to I \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \to Blurred \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

$$P_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} P_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Sobel

Sobel 算子是的基于一阶导数的边缘检测算子，是离散型的差分算子。对噪声具有一定平滑作用，能很好的消除噪声的影响。Sobel 算子对于像素的位置的影响进行加权，与 Prewitt 算子、Roberts算子相比因此效果更好。

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

Marr Hildreth

$$S = g * I$$

拉普拉斯(Lapla)

拉普拉斯算子类似于 2 阶的 sobel 算子，

$$\Delta^2 S =$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

代码实现一个卷积操作，

```
def conv(image,kernel):
    (iH, iW) = image.shape[:2]
    (kH, kW) = kernel.shape[:2]

    pad = (kW - 1) // 2
    image = cv2.copyMakeBorder(image, pad, pad, pad, pad,cv2.BORDER_REPLICATE)
    output = np.zeros((iH, iW), dtype="float32")
    for y in np.arange(pad, iH + pad):
        for x in np.arange(pad, iW + pad):
            roi = image[y - pad:y + pad + 1, x - pad:x + pad + 1]
            k = (roi * kernel).sum()
            output[y - pad, x - pad] = k
    output = rescale_intensity(output, in_range=(0, 255))
    output = (output * 255).astype("uint8")
    return output
```