昵称： nowgood
园龄： 2年9个月
粉丝： 56
关注： 11
+加关注

< 2020年3月 >

日	一	二	三	四	五	六
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

搜索

找找看

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

积分与排名

积分 - 107108
排名 - 5868

随笔分类 (108)

Algorithms(10)
C++(4)
DataBase(8)
Deep Learning(15)
Image Processing(3)
Interview(4)
Java Programming(2)
Linear Algebra and Probability Theory(10)
Linux(5)
Magic Toolkit(5)
Object Detection(2)
Paper Reading(11)
Python Programming(13)
Scala Programming(12)
TensorFlow(3)
Visualize(1)

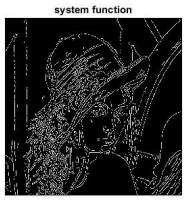
阅读排行榜

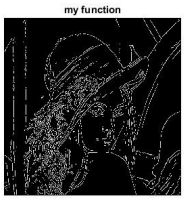
1. 10分钟轻松学会 Python turtle 绘图(106173)
2. 向量的L2范数求导(14268)
3. matplotlib 学习笔记(10201)
4. MarkDown 中使用 LaTeX 数学式(9673)
5. 动作识别调研(8719)
6. Huber Loss(8252)
7. tf.argmax 与 tf.arg_max(6389)
8. 使用 ffmpeg nginx rtmp 搭建实时流处理平台(5615)
9. Mac 远程桌面 ubuntu16.04 unity(5400)
10. 3步搞定博客园界面风格(5096)

评论排行榜

1. 10分钟轻松学会 Python turtle 绘图(15)
2. 3步搞定博客园界面风格(8)
3. Faster R-CNN(3)
4. 向量的L2范数求导(2)

边缘检测之Canny





该博客转载自 边缘检测之Canny，原博客格式清晰，数学分析到位，图文并茂，建议直接看，我这里就是做一个备份

1. 写在前面

最近在做边缘检测方面的一些工作，在网络上也找了很多有用的资料，感谢那些积极分享知识的先辈们，自己在理解Canny边缘检测算法的过程中也走了一些弯路，在编程实现的过程中，也遇到了一个让我怀疑人生的BUG（日了狗狗）。就此写下此文，作为后记，也希望此篇文章可以帮助那些在理解Canny算法的道路上暂入迷途的童鞋。废话少说，上干货。

2. Canny算法的发展史

Canny边缘检测于1986年由JOHN CANNY首次在论文《A Computational Approach to Edge Detection》中提出，就此拉开了Canny边缘检测算法的序幕。

Canny边缘检测是从不同视觉对象中提取有用的结构信息并大大减少要处理的数据量的一种技术，目前已广泛应用于各种计算机视觉系统。Canny发现，在不同视觉系统上对边缘检测的要求较为类似，因此，可以实现一种具有广泛应用意义的边缘检测技术。边缘检测的一般标准包括：

1）以低的错误率检测边缘，也即意味着需要尽可能准确的捕获图像中尽可能多的边缘。

2）检测到的边缘应精确定位在真实边缘的中心。

3）图像中给定的边缘应只被标记一次，并且在可能的情况下，图像的噪声不应产生假的边缘。

为了满足这些要求，Canny使用了变分法。Canny检测器中的最优函数使用四个指数项的和来描述，它可以由高斯函数的一阶导数来近似。

推荐排行榜

文章目录
1. 5搞定博客园界面风格(8)
2. 10分钟轻松学会 Python turtle 绘图(8)
3. MarkDown 中使用 LaTeX 数学式(4)
4. Scala 特质全面解析(2)
5. 边缘检测之Canny(2)
6. Fast RCNN 中的 Hard Negative Mining(1)
7. 函数式编程之foldLeftViaFoldRight(1)
8. matplotlib 学习笔记(1)
9. 向量的L2范数求导(1)
10. 隐马尔可夫模型后向算法详细推导(1)

以提供良好可靠检测的方法之一。由于它具有满足边缘检测的三个标准和实现过程简单的优势，成为边缘检测最流行的算法之一。

3. Canny算法的处理流程

Canny边缘检测算法可以分为以下5个步骤：

1）使用高斯滤波器，以平滑图像，滤除噪声。

2）计算图像中每个像素点的梯度强度和方向。

3）应用非极大值（Non-Maximum Suppression）抑制，以消除边缘检测带来的杂散响应。

4）应用双阈值（Double-Threshold）检测来确定真实的和潜在的边缘。

5）通过抑制孤立的弱边缘最终完成边缘检测。

下面详细介绍每一步的实现思路。

3.1 高斯平滑滤波

为了尽可能减少噪声对边缘检测结果的影响，所以必须滤除噪声以防止由噪声引起的错误检测。为了平滑图像，使用高斯滤波器与图像进行卷积，该步骤将平滑图像，以减少边缘检测器上明显的噪声影响。大小为(2k+1)x(2k+1)的高斯滤波器核的生成方程式由下式给出：

$$H_{ij}=\frac{1}{2\pi\sigma^2}\exp\left(-\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}\right);1\le i,j\le(2k+1) \quad (3\text{-}1)$$

下面是一个sigma = 1.4，尺寸为3x3的高斯卷积核的例子（需要注意归一化）：

$$H=\begin{bmatrix}0.0924 & 0.1192 & 0.0924\\ 0.1192 & 0.1538 & 0.1192\\ 0.0924 & 0.1192 & 0.0924\end{bmatrix}$$

若图像中一个3x3的窗口为A，要滤波的像素点为e，则经过高斯滤波之后，像素点e的亮度值为：

$$e=H*A=\begin{bmatrix}h_{11} & h_{12} & h_{13}\\ h_{21} & h_{22} & h_{23}\\ h_{31} & h_{32} & h_{33}\end{bmatrix}*\begin{bmatrix}a & b & c\\ d & e & f\\ g & h & i\end{bmatrix}=sum\left(\begin{bmatrix}a\times h_{11} & b\times h_{12} & c\times h_{13}\\ d\times h_{21} & e\times h_{22} & f\times h_{23}\\ g\times h_{31} & h\times h_{32} & i\times h_{33}\end{bmatrix}\right)$$

其中*为卷积符号，sum表示矩阵中所有元素相加求和。

重要的是需要理解，高斯卷积核大小的选择将影响Canny检测器的性能。尺寸越大，检测器对噪声的敏感度越低，但是边缘检测的定位误差也将略有增加。一般5x5是一个比较不错的trade off。

文章目录

图像中的边缘可以指向各个方向，因此Canny算法使用四个算子来检测图像中的水平、垂直和对角边缘。边缘检测的算子（如Roberts，Prewitt，Sobel等）返回水平Gx和垂直Gy方向的一阶导数值，由此便可以确定像素点的梯度G和方向theta 。

$$G = \sqrt{G_x^2 + G_y^2}$$
$$\theta = arc\tan\left(G_y / G_x\right) \quad (3\text{-}2)$$

其中G为梯度强度， theta表示梯度方向，arctan为反正切函数。下面以Sobel算子为例讲述如何计算梯度强度和方向。

x和y方向的Sobel算子分别为：

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

其中Sx表示x方向的Sobel算子，用于检测y方向的边缘； Sy表示y方向的Sobel算子，用于检测x方向的边缘（边缘方向和梯度方向垂直）。在直角坐标系中，Sobel算子的方向如下图所示。

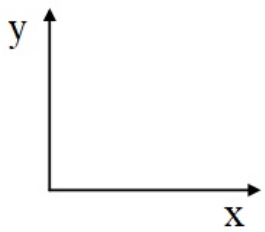


图3-1 Sobel算子的方向

若图像中一个3x3的窗口为A，要计算梯度的像素点为e，则和Sobel算子进行卷积之后，像素点e在x和y方向的梯度值分别为：

$$G_x = S_x * A = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = sum\left(\begin{bmatrix} -a & 0 & c \\ -2d & 0 & 2f \\ -g & 0 & i \end{bmatrix}\right)$$
$$G_y = S_y * A = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = sum\left(\begin{bmatrix} a & 2b & c \\ 0 & 0 & 0 \\ -g & -2h & -i \end{bmatrix}\right)$$

其中*为卷积符号，sum表示矩阵中所有元素相加求和。根据公式（3-2）便可以计算出像素点e的梯度和方向。

3.3 非极大值抑制

像进行梯度计算后，仅仅基于梯度值提取的边缘仍然很模糊。对于标准3，对边缘有且应当只有一个准确的响应。而非极大值抑制则可以帮助将局部最大值之外的所有梯度值抑制为0，对梯度图像中每个像素进行非极大值抑制的算法是：

文章目录

1）将当前像素的梯度强度与沿正负梯度方向上的两个像素进行比较。

2）如果当前像素的梯度强度与另外两个像素相比最大，则该像素点保留为边缘点，否则该像素点将被抑制。

通常为了更加精确的计算，在跨越梯度方向的两个相邻像素之间使用线性插值来得到要比较的像素梯度，现举例如下：

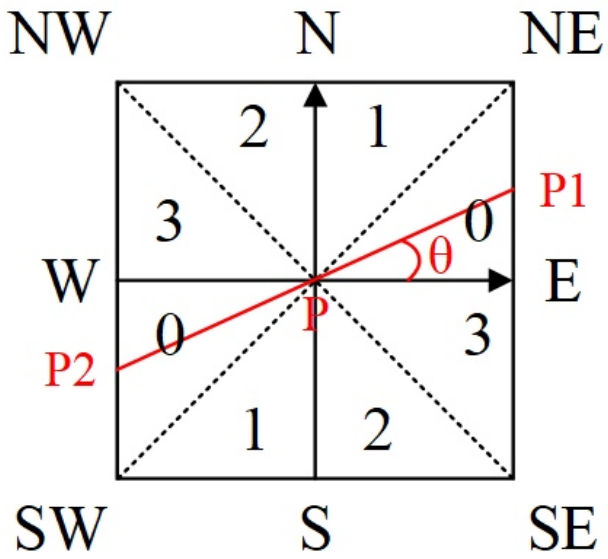


图3-2 梯度方向分割

如图3-2所示，将梯度分为8个方向，分别为E、NE、N、NW、W、SW、S、SE，其中0代表00~45o,1代表450~90o，2代表-900~-45o，3代表-450~0o。像素点P的梯度方向为theta，则像素点P1和P2的梯度线性插值为：

$$\tan(\theta) = G_y / G_x$$

$$G_{p1} = (1 - \tan(\theta)) \times E + \tan(\theta) \times NE$$

$$G_{p2} = (1 - \tan(\theta)) \times W + \tan(\theta) \times SW$$

因此非极大值抑制的伪代码描写如下：

$$\begin{aligned} &if \;\; G_p \geq G_{p1} \;\; and \;\; G_p \geq G_{p2} \\ &\quad G_p \; may \; be \; an \; edge \\ &else \\ &\quad G_p \;\; should \;\; be \;\; \text{sup} \; pressed \end{aligned}$$

需要注意的是，如何标志方向并不重要，重要的是梯度方向的计算要和梯度算子的选取保持一致。

文章目录

在施加非极大值抑制之后，剩余的像素可以更准确地表示图像中的实际边缘。然而，仍然存在由于噪声和颜色变化引起的一些边缘像素。为了解决这些杂散响应，必须用弱梯度值过滤边缘像素，并保留具有高梯度值的边缘像素，可以通过选择高低阈值来实现。如果边缘像素的梯度值高于高阈值，则将其标记为强边缘像素；如果边缘像素的梯度值小于高阈值并且大于低阈值，则将其标记为弱边缘像素；如果边缘像素的梯度值小于低阈值，则会被抑制。阈值的选择取决于给定输入图像的内容。

双阈值检测的伪代码描写如下：

$$
\begin{aligned}
&if\ \ G_p \geq HighThreshold \\
&\quad G_p\ is\ an\ strong\ edge \\
&else\ \ if\ \ G_p \geq LowThreshold \\
&\quad G_p\ is\ an\ weak\ edge \\
&else \\
&\quad G_p\ \ should\ \ be\ \ \sup pressed
\end{aligned}
$$

3.5 抑制孤立低阈值点

到目前为止，被划分为强边缘的像素点已经被确定为边缘，因为它们是从图像中的真实边缘中提取出来的。然而，对于弱边缘像素，将会有一些争论，因为这些像素可以从真实边缘提取也可以是因噪声或颜色变化引起的。为了获得准确的结果，应该抑制由后者引起的弱边缘。通常，由真实边缘引起的弱边缘像素将连接到强边缘像素，而噪声响应未连接。为了跟踪边缘连接，通过查看弱边缘像素及其8个邻域像素，只要其中一个为强边缘像素，则该弱边缘点就可以保留为真实的边缘。

抑制孤立边缘点的伪代码描述如下：

$$
\begin{aligned}
&if\ \ G_p == LowThreshold\ \ and\ \ G_p\ \ connected\ to\ a\ strong\ edge\ pixel \\
&\quad G_p\ is\ an\ strong\ edge \\
&else \\
&\quad G_p\ \ should\ \ be\ \ \sup pressed
\end{aligned}
$$

4 总结

通过以上5个步骤即可完成基于Canny算法的边缘提取，图5-1是该算法的检测效果图，希望对大家有所帮助。

文章目录

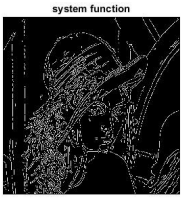


图5-1 Canny边缘检测效果

分类: Image Processing

标签: 计算机视觉, canny, 边缘检测

nowgood
关注 - 11
粉丝 - 56
+加关注

2 0

« 上一篇: 神经网络优化算法
» 下一篇: Python 常用排序函数

posted @ 2018-09-10 08:52 nowgood 阅读(3960) 评论(1) 编辑 收藏

评论列表

#1楼 2020-01-27 02:14 H_Li98

写得非常清楚

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册， 访问 网站首页。