运用 Freeman 准则的直线检测算法

尚振宏[1,2] 刘明业[3]

[1](北京理工大学 ASIC 研究所 北京 100081)

[2](昆明理工大学信息工程与自动化学院 昆明 650051)

[3](厦门大学计算机与信息工程学院 厦门 361005)

(shangzhenhong@sina.com)

摘 要 提出了一种简单而高效的在二值图像中检测目标物体直线边界的算法. 基于 Freeman 提出的关于数字直线的准则和数字直线的特征,得出线段元是数字直线的组成部分这一性质. 基于该性质,该算法以线段元为基本单位进行直线的构造,从而能高效、准确地检测出图像中物体边界中的直线. 此外,该算法还可用于检测二值图像中物体边界的拐角.

关键词 直线检测 边界跟踪 链码

中图法分类号 TP391.4

Line Detection Algorithm Using Freeman Criteria

Shang Zhenhong[1,2] Liu Mingye[3]

[1](*ASIC Research Center of Beijing Institute of Technology*, *Beijing* 100081)

[2](*School of Information Engineering and Automation*, *Kunming University Science and Technoly*, *Kunming* 650051)

[3](*School of Computer and Information Engineering*, *Xiamen University*, *Xiamen* 361005)

Abstract This paper proposes a simple and efficient algorithm to detect line edge of objects in a binary image. Based on the criteria and characteristic of digital line suggested by Freeman, we derived that digital line is composed of sets of line segment cell. Derived from this property of line cell, in the algorithm proposed, line cells are used for connection to form line segment. It makes the algorithm very efficient and precise. This algorithm can also be used to detect corner of objects in binary images.

Key words line detection; edge tracking; chain code

1 引 言

在计算机视觉领域,直线或线段的检测是一个非常重要的过程. 很多物体具有直线轮廓,检测并定位这些直线轮廓,从而定位目标,可以为进一步进行目标识别和分析提供条件. Hough 变换是一个著名的直线检测算法[1],该算法将图像空间(直角坐标)中的每一点映射为参数空间(极坐标)中的一条正弦型曲线,最终通过检测这些正弦型曲线交点的峰值定位出图像中的直线. 由于 Hough 变换是对图像中所有的点进行处理,取最后的综合效果,因此该算法具有抗干扰能力强并宜于并行实现的特点. 但它是一种穷尽式搜索,其计算复杂度和空间复杂度都很高. 多年来,人们对直线检测问题展开了深入的研究,并提出了许多 Hough 变换的改进算法[2],希望能大幅度地提高该算法的速度,但迄今为止,未见取得重大突破.

Yuan 等提出了一种在链码中检测直线的算法[3]. 该算法从起始链码开始,对每一链码确定一个直线穿行区域以及两条用于确定下一链码是否属于同一直线的上下边界线. 如果下一链码位于上下边界线之内,则该链码与上一链码属于同一直线;否则该链码属于另一直线. 由于该算法仅对目标的边

收稿日期:2003-07-22;修回日期:2003-11-12

界链码进行处理,因此其算法的复杂度较小,为 $O(n)$,其中 n 为边界链码的个数. 但该算法在跟踪得到每一链码时需确定下一直线链码的存在范围,这是一个比较耗时的过程,利用数字直线的特征可以简化该过程.

计算机处理的图像是经过采样、量化等数字过程后形成的离散图像,离散空间中的直线呈现出连续空间的直线所不具备的一些特征. Freeman 总结了这些特征并提出了数字直线的链码应遵循的三条准则[4](简称 Freeman 准则):(1)一条数字直线的 8 邻域链码中最多包括两个方向,其中一个为主方向,它是决定直线方向的主要因素;(2)这两个方向的链码值相差为 1(mod 8);(3)主方向上链码值相同的连续像素组成一个线段子元,除去第一个和最后一个线段子元,其余各线段子元的长度至多相差一个像素. 基于 Freeman 准则,Chan 等提出一个直线检测算法[5],该算法跟踪线段子元,并根据两相邻线段子元间的偏转角度确定这两线段子元是否相似,若相似则连接两线段子元. 该算法的计算复杂度为 $O(n)$,n 为目标边界像素的数目. 虽然该算法与 Yuan 等算法的计算复杂度相同,但该算法在跟踪得到一个线段子元后才进行偏转角度的计算,因此,该算法比 Yuan 等算法更高效. 但同时我们注意到,该算法没有充分利用 Freeman 准则,并非最优,留下了改进的空间.

充分利用 Freeman 准则,本文提出一个高效的直线检测算法. 与已有算法不同,本文算法跟踪得到线段子元,但并不计算线段子元间的角度偏差,而直接利用 Freeman 准则进行判断,将线段子元合并为更长的单位——线段元,然后再计算线段元与线段间角度的偏差,从而判断线段元是否属于线段. 因此本文算法具有更高的实时性.

2 算法描述

定义 1. 线段子元. 使用 8 邻域 *Freeman* 边界链码(如图 1 所示)表示边界时,链码值相同且为偶数的像素的集合称为一个线段子元,线段子元的前一像素和后一像素的链码值均与该线段子元中的链码值不同. 图 2 中的数字线段由 4 个线段子元(a,b,c 和 d)构成,线段子元 a 和 d 的长度均为 5 个像素,线段子元 b 和 c 的长度分别为 9 和 10 个像素.

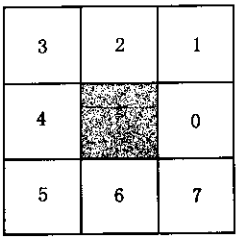


图 1 8 邻域链码

定义 2. 线段元. 线段元由一个或多个线段子元构成. 线段元中各链码至多有两个取值,这两个不同取值的链码的方向相差 45°. 为了满足 Freeman 准则 3,构成线段元的线段子元间的长度至多相差一个像素. 图 2 中的线段中包含三个线段元(1,2 和 3),线段元 2 由线段子元 b 和 c 构成.

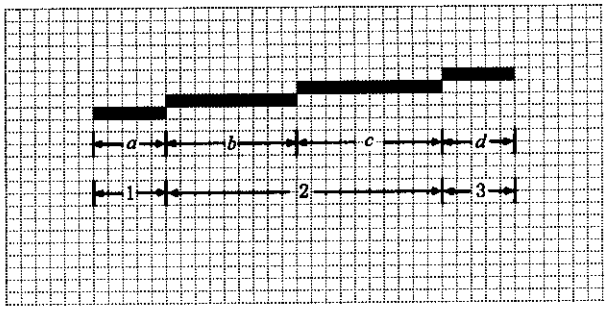


图 2 线段子元与线段元

由以上定义可知,线段子元是线段元的子集,一个线段元可能仅由一个线段子元构成. 比如图 2 中的线段子元 a 同时也是线段元 1.

性质. 线段元 ls 是构成线段(或直线) l 的子集.

证明. 由 Freeman 准则可知,在线段元 ls 的起始端点之前和终止端点之后分别添加 0 个或多个线段子元,必可构建一条线段(或直线) l. 因此,线段元 ls 为线段(或直线) l 的一部分,即线段元 ls 是构成线段(或直线) l 的一个子集. 证毕.

基于以上性质,本文提出新的直线检测算法. 首先根据定义 2 找到图像中物体边界的线段元,然后根据线段元与直线的夹角确定该线段元是否属于该直线. 由于算法中是以线段元为单位进行判断计算的,因此该算法比前面所述的几种算法更为高效.

本算法的输入为二值图像,输出为图像中物体边界的各条线段. 整个算法流程如下所述,其中,*Line_Sub_Cell*、*Line_Cell* 和 *Line_Segment* 分别表示存储线段子元、线段元和线段的结构.

Step1. 从扫描二值图像,定位边界跟踪起点 P_i,$i=0$. 将 P_i 作为第一个像素保存到线段子元 *Line_Sub_Cell* 中.

Step2. $i=i+1$,利用文献[6]中提出的边界跟踪算法顺时针(或逆时针)跟踪得到下一边界点 P_i,若跟踪回到起始点,则算法结束 否则,根据定义 1 确定 P_i 是否属于当前线段子元 *Line_Sub_Cell*. 若属于,则将 P_i 保存到 *Line_Sub_Cell* 中,转 Step2;否则,当前线段子元 *Line_Sub_Cell* 即为一完整的线段子元,执行 Step3.

Step3. 若线段元 *Line_Cell* 中无线段子元,则将 *Line_Sub_Cell* 作为 *Line_Cell* 的第一线段子元加入 *Line_Cell*;否则,根据定义 2 判断线段子元 *Line_Sub_Cell* 是否属于线段元 *Line_Cell*. 若属于,将 *Line_Sub_Cell* 加入 *Line_Cell*,然后将 *Line_Sub_Cell* 置空并将 P_i 加入 *Line_Sub_Cell* 中,转 Step2;否则,*Line_Cell* 为一完整的线段元,执行 Step4.

Step4. 若当前线段 *Line_Segment* 中无线段元,则将 *Line_Cell* 作为 *Line_Segment* 的第一线段元加入 *Line_Segment*;否则根据下面的准则判断 *Line_Cell* 是否属于 *Line_Segment*. 若不属于,则 *Line_Segment* 为一完整直线段,保存该直线段并置空 *Line_Segment*. 然后,无论 *Line_Cell* 是否属于 *Line_Segment* 均将 *Line_Cell* 加入 *Line_Segment*,并分别用当前的线段子元 *Line_Sub_Cell* 和点 P_i 初始化线段元 *Line_Cell* 和线段子元 *Line_Sub_Cell* 并转回 Step2.

判断线段元 *Line_Cell* 是否属于线段 *Line_Segment* 的准则:

设线段元 *Line_Cell* 中起始像素点的坐标为 (X_{LCS}, Y_{LCS}),终止像素点的坐标为 (X_{LCE}, Y_{LCE}),*Line_Cell* 中像素的个数为 M;线段 *Line_Segment* 中起始像素点的坐标为 (X_{LSS}, Y_{LSS}),终止像素点的坐标为 (X_{LSE}, Y_{LSE}),*Line_Segment* 中像素的个数为 N,则线段元及线段偏转角分别为

$$\theta_{LC} = \arctan\frac{Y_{LCE} - Y_{LSS}}{X_{LCE} - X_{LSS}},$$

$$\theta_{LS} = \arctan\frac{Y_{LSE} - Y_{LSS}}{X_{LSE} - X_{LSS}}.$$

若

$$|\theta_{LC} - \theta_{LS}| \leqslant \frac{\alpha}{M + N} \tag{1}$$

则线段元 *Line_Cell* 属于线段 *Line_Segment* 的一部分. 其中 α 为一阈值.

这里用 $\alpha/(M+N)$ 的值来判断线段元偏转角 θ_{LC} 和线段偏转角 θ_{LS} 是否接近的原因在于:对于一条数字线段,当线段较短时线段元与线段偏转角间的偏差较大,而随着线段长度的增加,线段元与线段偏转角间的差异将趋于 0[5]. 由式(1)可知,阈值 α 决定了算法的精度. α 较小时,检测出的线段 *Line_Segment* 中包含的线段元 *Line_Cell* 较少,检测的精度较高;当 α 趋于 0 时,每条线段蜕变为一个线段元;α 较大时,线段 *Line_Segment* 中包含的 *Line_Cell* 较多,检测的精度较低,但抗噪声能力较强. 因此,应根据实际应用需要确定 α 值,在实际中我们选 $\pi/4 \leqslant \alpha \leqslant \pi$.

由于跟踪起点可能位于一条线段中的某一位置,因此对于图像中的某一直线边界,本算法处理后可能得到两条较短的线段,合并这两条较短的线段便可构成图像中的直线边界. 在本算法结束后,如有需要可进行一个线段合并的计算. 线段合并的方法比较简单,即判断两线段的端点是否相邻(或接近),并且两线段的偏转角是否接近. 若满足上述条件即可将两线段连接起来,形成一条较长的线段. 可以看出,线段合并的过程也具有抑止噪声的功能. 在本算法完成后可根据检测到的线段的偏转角或长度,根据不同的应用需要选择出所需的线段,从而达到定位目标物体的目的.

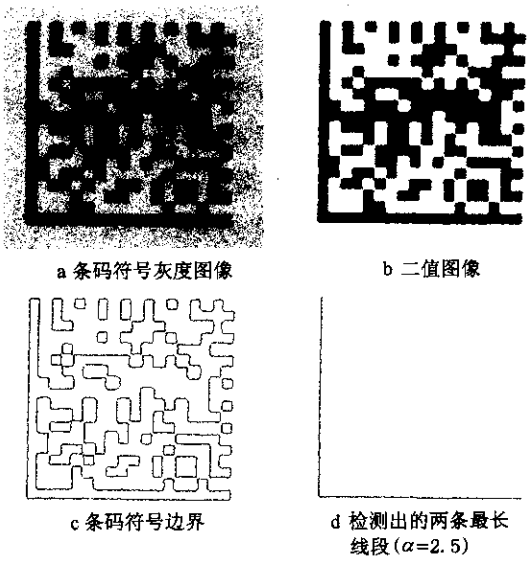
若需检测图像中多个物体的直线边界,在实现本算法时,可引入一标记图像[7],该标记图像和二值图像具有相同的尺寸. 当跟踪得到一线段后,即在标记图像中相应位置处用一随线段数目递增的标号标记该线段上的所有像素. 在计算跟踪起点时,若某一像素已被标记过,则可跳过而寻找其他的跟踪起点,从而加快处理速度. 除了检测图像中闭合边界中的直线外,利用标记图像并修改算法的结束条件. 本文算法也适合检测非闭合边界中的直线,此时跟踪一个像素便在标记图像中相应位置处作一标记. 在跟踪过程中若某一像素已被标记过或跟踪到达了图像的边界,则边界跟踪结束.

从对算法的描述可看出,本文算法的计算复杂度为 $O(n)$,n 为图像中物体边界像素的个数. 算法的 Step2,Step3 仅需根据线段子元和线段元的定义进行判断即可,无需进行偏转角的计算,而该过程通常是一相对耗时的过程;在算法的 Step4,才以线段元为单位进行偏转角差别的计算和判断. 因此本文算法具有简单、高效的特点.

3　实验结果

在实验 1 中,运用本文算法在 Data Matrix 二维条码符号的二值图像中检测‘直角’型定位图形. 图 3a 是用 CMOS 摄像头实际拍摄得到的尺寸为 400 像素×388 像素的 Data Matrix 二维条码灰度图像,并在该图像中加入了 5% 的高斯白噪声. 该灰度图像二值化的结果如图 3b 所示. 本文算法跟踪该二值图像中条码符号的边界(如图 3c 所示),并从边界像素中检测出两条最长的线段,这两条线段即为 Data Matrix 二维条码符号的‘直角’型定位图形(如图 3d

所示). 该‘直角’型定位图形是 Data Matrix 二维条码中的特征图形,高效、准确地定位出该‘直角’型定位图形是 Data Matrix 二维条码识读的关键. 后续译码算法根据该定位图形的两直角边定位符号并确定条码符号的参数. 在上述直线检测过程后进行了线段合并的计算. 检测的结果未用直线拟合条码符号的边界,而是精确定位出边界各像素的位置. 这是因为在后续的识别过程中需要这些边界像素的位置数据. 表 1 为三种直线检测算法在检测图3a中“直角”型探测图形时运行效率的比较. 从实验结果可以看出,Hough 变换是一种实时性较差的直线检测算法. 在实验环境 A 和实验环境 B 下,本文算法的效率均优于参考文献[5]的算法;在实验环境 B 下,表 1 中前两种算法效率下降的程度明显超过了本文算法. 这主要是因为,MC9328MX1 是一款以 ARM9 为核的 CPU,该系列的 CPU 在硬件上不支持浮点运算(这在嵌入式系统较常见). 本文算法以线段元为单位进行线段的检测,有效地减少了浮点运算,因而得到了较好的实验结果.



a 条码符号灰度图像　b 二值图像　c 条码符号边界　d 检测出的两条最长线段(α=2.5)

图 3　实验 1 检测 Data Matrix 条码探测图形

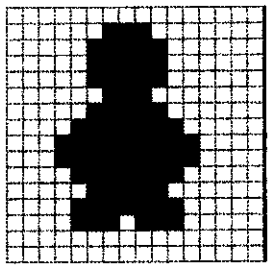
表 1　三种直线检测算法效率的比较

算法	时间	
	实验环境 A*	实验环境 B**
Hough 变换[8]	2230	13815000
文献[5]算法	70	5320
本文算法	40	1660

* 操作系统:Redhat Linux 9.0,CPU:PⅢ800,RAM:128MB

** 操作系统:eLinux BSP0.3.4,CPU:MC9328MX1,RAM:64MB

在实验 2 中,运用本文算法检测如图 4a 所示尺寸为 8×8 像素的图像中‘人’型的物体的直线边界.



a 二值图像

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	3	3	3	0	0	0	0
0	0	0	2	0	0	0	4	0	0	0
0	0	0	2	0	0	0	4	0	0	0
0	0	0	2	0	0	0	4	0	0	0
0	0	0	0	1	0	5	0	0	0	0
0	0	0	1	0	0	0	5	0	0	0
0	0	1	0	0	0	0	0	5	0	0
0	1	0	0	0	0	0	0	0	5	0
0	12	0	0	0	0	0	0	0	5	0
0	0	12	0	0	0	0	0	6	0	0
0	0	0	11	0	0	0	6	0	0	0
0	0	11	0	0	9	0	0	7	0	0
0	0	10	10	10	0	8	8	7	0	0
0	0	0	0	0	0	0	0	0	0	0

b $\alpha=1$ 时的标记图像

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	3	3	3	0	0	0	0
0	0	0	2	0	0	0	4	0	0	0
0	0	0	2	0	0	0	4	0	0	0
0	0	0	2	0	0	0	4	0	0	0
0	0	0	0	1	0	4	0	0	0	0
0	0	0	1	0	0	0	4	0	0	0
0	0	1	0	0	0	0	0	4	0	0
0	1	0	0	0	0	0	0	0	4	0
0	10	0	0	0	0	0	0	0	4	0
0	0	10	0	0	0	0	0	5	0	0
0	0	0	9	0	0	0	5	0	0	0
0	0	11	0	0	7	0	0	6	0	0
0	0	8	8	8	0	7	7	6	0	0
0	0	0	0	0	0	0	0	0	0	0

c $\alpha=2.5$ 时的标记图像

图 4　α 值对检测结果的影响

该实验显示了阈值 α 是直线近似程度的决定因素. 在实验 2 中,我们引入了标记图像来演示阈值 α 取不同值时,直线检测的结果. 在图4b和图4c中,每一像素上的数值为该像素的标记值,所有标记值为 n 的像素组成了检测出的第 n 条线段. 当 α =

1 时,共检测出了 12 条线段,如图4b所示. 若 α 取更小的值,直线检测的结果不会再有变化,这是因为在图4b中每一被标记的线段实际上仅是一线段元. 当$\alpha=2.5$ 时,共检测出了 10 条线段,如图4c所示,其中线段 4 由两个线段元构成. 该实验显示出当 α 较小时,本文算法可精确检测出物体的直线边界;当 α 较大时,本文算法将角度偏差在一定范围内的曲线近似探测为直线,因此具有较强的抗干扰能力.

4 结　论

基于 Freeman 准则,本文提出了一种检测直线边界的算法,该算法以线段元为单位进行线段的构造,具有较高的实时性. 与许多直线检测算法不同,本文算法无需对目标图像进行边缘检测处理,而直接处理目标的二值图像,在跟踪目标物体的边界时完成直线的检测,从而使得本文算法的检测结果能精确定位到目标物体边界. 此外,本文算法还可用于检测二值图像中物体边界的拐角.

参 考 文 献

[1] Hough P V C. Methods and means for recognizing complex patterns [P]. USA, United States Patent, 3069654, 1962

[2] Kassim A A, Tan T, Tan K H. A comparative study of efficient generalized Hough transform techniques [J]. Image and Vision Computing, 1999, 17(10): 737～748

[3] Yuan Jianxing, Suen Ching Y. An optimal algorithm for detecting straight lines in chain codes [A]. In: Proceedings of the 11th IAPR International Conference on Pattern Recognition, Hague, 1992. 692～695

[4] Freeman H. Boundary encoding and processing [A]. In: Picture Processing and Psychopictorics [C]. New York: Academic, 1970. 241～266

[5] Chan T S, Yip R K K. Line detection algorithm [A]. In: Proceedings of the 13th International Conference on Pattern Recognition, Vienna, 1996. 126～130

[6] Shi Ce. A discussion of a fast algorithm for boundary tracing [J]. Mini-Micro Systems, 2000, 21(6): 641～645 (in Chinese)
(史　册. 对一种边缘跟踪算法的讨论[J]. 小型微型计算机系统, 2000, 21(6): 641～645)

[7] Venkateswar V, Chellappa R. Extraction of straight lines in aerial images [J]. IEEE Transactions on Patten Analysis and Machine Intelligence, 1992, 14(11): 1111～1114

[8] Kalviainen H, Hirvonen P, Oja E. Houghtool—A software package for use of Hough transform [J]. Pattern Recognition Letters, 1996, 17(8): 889～897

尚振宏　男,1975 年生,博士,主要研究方向为计算机图形与图像处理、计算机视觉.

刘明业　男,1934 年生,教授,博士生导师,主要研究方向为多媒体信息处理、ASIC 设计、EDA 技术.