二值图像中拐点的实时检测算法

尚振宏[1),2)] 刘明业[3)]

[1)](北京理工大学 ASIC 研究所,北京 100081) [2)](昆明理工大学计算机系,昆明 650051)

[3)](厦门大学计算机与信息工程学院,厦门 361005)

摘 要 鉴于数字图像中的拐点通常成为重要的信息载体,因此准确、稳定和实时地检测出拐点便成为拐点检测算法面临的主要问题,针对该问题,提出了一种新的二值图像中拐点的实时检测算法。该算法与传统基于边界链码的拐点检测算法不同,其是首先构建像素的 $k(k>8)$ 邻域,并将图像中物体的边界表示为 k 邻域链码;然后根据曲率定义的差分形式计算各边界点处的曲率;最后通过检测曲率直方图的局部峰值精确定位出拐点,并利用拐角内部像素的颜色统计信息迅速判断出拐点的凸凹性.为验证该算法的效果,给出了该算法与 4 种已有算法的对比实验.结果表明,该算法不仅稳定性、准确性较高,而且算法简单,实时性强,并适合于嵌入式计算环境。

关键词 拐点检测 链码 边界跟踪

中图法分类号: TP391.4 **文献标识码**: A **文章编号**: 1006-8961(2005)03-0295-06

Real-Time Corner Detection in Binary Image

SHANG Zhen-hong[1)], LIU Ming-ye[2)]

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

[2)] (*Department of Computer Science and Technology, Kunming University of Science and Technology, Kunming* 650051)

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

Abstract Presents a new real-time corner detection algorithm. Corners are important information carriers in object recognition. Accurate, stable and fast detecting corners in digital image are common problems facing to corner detectors. Aiming at these problems and different from traditional corner detection algorithms, based on chain code, the algorithm constructs $k(k>8)$ neighborhood chain codes of pixels and uses these chain codes to describe contours. Based on the differential definition of curvature, a curvature function is derived from k neighborhood chain codes. Corners are detected as those contour pixels, whose curvature the is largest in a lobe of contour curvature histogram. Convex and concave corners can be differentiated quickly by checking color attributes of pixels between corner edges. To validate the algorithm, tests comparing the new algorithm to 4 corner detection algorithms are given. The results show the new algorithm is not only accurate and stable, but also simple and fast, which make the algorithm suitable for the embedded computation environment.

Keywords corner detection, chain code, edge tracking

1 引 言

由于数字图像中拐点通常传递了非常重要的信息,而且这些信息被用于定位目标以及进一步识别目标,因此拐点检测成为计算机视觉中的一个研究重点。近年来,许多学者就这一问题展开了广泛研究。根据算法处理的像素集的不同,拐点检测算法可分为以下两类:(1)先利用一掩模算子来处理图像中的每一像素及其邻域像素,然后根据处理结果选择出拐点[1,2];(2)先处理图像中物体的边界像素,再通过寻找边界上曲率的局部最大值来定位拐点[3~6]。实践证明,第 2 类算法与第 1 类算法相比,具有以下优点:(1)由于第 2 类算法仅对物体边界上的像素进行计

收稿日期:2004-01-08;**改回日期**:2004-08-23

第一作者简介:尚振宏(1975~),男,讲师,2004 年 9 月毕业于北京理工大学计算机系获工学博士学位,现为昆明理工大学计算机系讲师。主要研究方向为计算机图形与图像处理、计算机视觉、2 维条码技术。E-mail:shangzhenhong@sina.com

算，因此其具有较低的计算复杂度，不但实时性较高，并且通常具有算法简单、易于实现的特点；(2)由于在实施第 2 类算法前，通常先进行边界跟踪，因而使得在最终检测出的拐点中包含了拐点顺序这一信息。该信息对于一些应用非常重要(如，在 2 维条码识别中就利用该信息来确定条码符号的旋转角度)，而在第 1 类算法中则较难获取这类信息。

基于上述原因，本文主要讨论第 2 类算法。文献[3]中，Rosenfeld 与 Johnston 提出 k 余弦算法，即利用边界点处内角的余弦值来估算曲率，本文将其标识为 RJ 算法。Rosenfeld 与 Weszka 改进了 RJ 算法，用各点处边界邻域中内角的平均余弦值来平滑曲率[4]，这虽提高了拐点识别的准确性，但却极大地降低了算法的实时性。本文将该算法标识为 RW 算法。文献[5]中，Freeman 与 Davis 首先利用一条连接物体边界上由 s 段链码构成的曲线段的两个端点的移动直线段来平滑物体边界，然后根据直线段两次移动的偏转角度来近似计算边界各点的曲率，但该算法中，参数 s 的选择对拐点检测结果的影响非常大，本文将此算法标识为 FD 算法。Beus 和 Ti 又提出了 FD 算法的改进算法[6]，本文将其标识为 BT 算法。与 RW 算法类似，该算法是通过对用 FD 算法计算得到的曲率进行局部平均来改善算法的稳定性。

上述算法的共同缺点在于：(1)由于计算得到的拐点处的曲率与非拐点处的曲率差别较小，因此算法的稳定性和准确性对图像中物体的形状及尺寸变化较敏感，且较易受到噪声的干扰；(2)由于这些算法中包含较多浮点运算，在仅支持整数运算的嵌入式环境中实现时，这些算法的实时性并不理想。针对这些问题，本文提出了基于 k(k = 16,24,32,…)邻域链码的实时拐角检测算法。该算法虽属于第 2 类拐点检测算法，但与已有基于链码的算法不同。即这些已有的算法均基于 8 邻域链码，而 8 邻域链码量化精度较低(将 2π 平面空间量化为 8 个方向)。一般情况下，相邻链码的差值不能准确反映曲率变化，需引入其他机制，以抑制边界噪声，而该过程又往往会引入浮点运算和乘法(或除法)等高开销运算(如 RJ 算法和 BT 算法)。相对于 8 邻域链码而言，由于利用 k 邻域链码既增加了对边界的平滑，又提高了方向量化精度，从而既可直接利用 k 邻域链码来较准确地计算出各边界点的曲率，以有效地避免浮点数运算，又可提高拐点与非拐点间曲率的差别，以提高算法的稳定性和准确性。实验结果表明，该算法不仅具有较高的实时性和准确性，而且阈值选择空间大、算法稳定、抗噪声能力强。

2 算 法

2.1 k 邻域链码

在本算法描述中，设图像中物体的闭合边界由 n 个像素 $p_i(i=1,2,\cdots,n)$ 构成；$c_{(k,i)}$ 和 e_i 分别表示第 i 个边界像素处的 k 邻域链码值和曲率。由于边界闭合，因此 $p_i = p_{i\pm n}$；$c_{(k,i)} = c_{(k,i\pm n)}$；$e_i = e_{i\pm n}$。

像素 p_i 的 k 邻域即是将以 p_i 为中心的 2π 平面空间量化为 k 个方向，每一方向分别用数字 0，1，…，$k-1$ 标识。例如，图 1 中心的黑色模块表示像素，其外围的第 1、2 和 3 圈模块分别为该像素的 8、16 和 24 邻域。在某一 k 邻域中，由于像素 $p_{(i+k/8)}$ 的位置可用相对于 p_i 的链码 $c_{(k,i+k/8)}$ 表示出，因而可用 k 邻域链码表示出物体的边界，例如，图 2 为跟踪得到的三角形图案的闭合边界，其 16 邻域链码为：13，13，13，0，2，2，2，1，1，2，1，1，8，7，7，8，7，7，8，8，7，7，8，8，10，13，13，13，14，13。

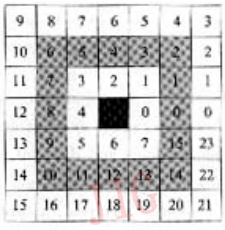


图 1 像素的 8、16、24 方向邻域

Fig. 1 8, 16 and 24 neighborhood of pixels

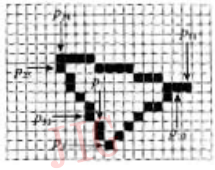


图 2 边界像素

Fig. 2 Contour pixels

存在以下两种特殊情况：

(1)像素 $p_{(i+k/8)}$ 不位于 p_i 的 k 邻域内，但位于 p_i 的 $k-8m$ 邻域内($m=1,2,3,\cdots$，且 $k-8m\geq 8$)。由于像素 $p_{(i+k/8)}$ 的 k 邻域链码值与其 $k-8m$ 邻域链码值存在近似的 $k/(k-8m)$ 倍关系，由此可近似求得像素 $p_{(i+k/8)}$ 的 k 邻域链码值

$$c_{(k,i+k/8)} \approx c_{(k-8m,i+k/8)} \times [k/(k-8m)]$$

例如,图 2 中由于像素 p_{25} 并不位于 p_{23} 的任一 16 邻域位置,从而不能直接得到链码值 $c_{(16,25)}$,但由于 p_{25} 处于 p_{23} 的 8 邻域内,且 $c_{(8,25)}=5$,从而可近似得到 $c_{(16,25)} \approx 10$。

(2)若像素 $p_{(i+k/8)}$ 不位于 p_i 的任一邻域内,则此时 $p_{(i+k/8)}$ 与 p_i 实际上为同一像素,并且意味着 $p_{(i+k/16)}$ 为拐点,且拐角为 π,例如图 2 中,p_{12} 和 p_{10} 实际上为同一像素,且该像素与一线段端点 p_{11} 相邻。在这种情况下,可分别对像素 $p_{(i+k/8)}$ 和 p_i 做一标记,并在下面的曲率计算中,利用该标记来突出像素 $p_{(i+k/16)}$ 处曲率的变化。由 2.2 节可知,由于此时 $c_{(k,i+k/8)}$ 不参与曲率计算,因此可将 $c_{(k,i+k/8)}$ 设为任意值,不妨令 $c_{(k,i+k/8)}=c_{(k,i+k/8-1)}$。

2.2　曲率计算

连续空间中某一边界点的曲率 e 定义为

$$e = \lim_{\Delta r \to 0} \left| \frac{\Delta \theta}{\Delta r} \right|$$

其中,$\Delta\theta$ 为该点处切线倾角的变化,Δr 为该点处弧长的变化。

由上述链码表示过程可知,$c_{(k,i+k/8)}-c_{(k,i)}$ 即为物体边界点 p_i 处切线倾角变化的差分表示。在离散空间中,若用差分代替微分,并注意在 k 邻域链码的第 2 种特殊情况中所做的标记,则边界点 p_i 处的曲率 e_i 可通过式(1)~式(3)计算得到

$$\theta_i = \begin{cases} |c_{(k,i+k/8)} - c_{(k,i)}| & p_i \text{ 未标记} \\ 0 & p_i \text{ 已标记} \end{cases} \tag{1}$$

其中,θ_i 为物体边界点 p_i 处切线倾角变化的差分表示。由于拐角的范围为 $[0,\pi]$,因此可初步得到 p_i 点处的曲率

$$\varphi_i = \begin{cases} \theta_i & \theta_i \leqslant k/2 \\ k-\theta_i & \theta_i > k/2 \end{cases} \tag{2}$$

由于拐点及其附近点处初步得到的曲率 φ_i 往往具有相对较大值,并且在拐点检测中,仅需比较各边界点处曲率的相对大小,因此可通过式(3)进一步扩大拐点与非拐点处曲率的差别,并完成曲率计算:

$$e_i = \varphi_i \sum_{j=-1}^{1} \varphi_{i+j} \tag{3}$$

2.3　算法描述

本算法首先利用 k 邻域链码求得二值图像中物体边界各点的曲率,然后通过寻找曲率的局部峰值检测出拐点,并区分出各拐点的凸凹性。具体算法描述如下:

(1)选取 k 值　由 2.2 节可知,k 值越大,不仅对边界点的平滑作用越强,而且曲率计算的抗噪声性能也越好,但由于当链码的编码步幅($k/8$)的 2 倍大于拐点间最短距离 d 时,则会因曲率直方图的波瓣底部有部分重合而干扰拐点检测,因此应将 k 值限制为 $k \leqslant 4d$,并可用下式计算 k 值:

$$k = \lfloor d/2 \rfloor \times 8$$

式中,$\lfloor\ \rfloor$ 表示向下取整,在应用中可根据识别对象的特性给出 d 值;

(2)跟踪二值图像中物体的边界,并用 k 邻域链码表示边界各像素;

(3)按式(3)计算边界各点的曲率　例如,图 3(b)~3(d)为由图 2 中各边界点的 16 邻域~32 邻域曲率构成的直方图;

(4)计算曲率局部峰值的位置　顺序扫描各点处的曲率 $e_i(i=1,2,\cdots,n)$,若 $e_{i-1}<t$,且 $e_i \geqslant t$,则 i 为波瓣的起始点,记为 i_{start};随后继续扫描,找到满足 $e_i \geqslant t$,且 $e_{i+1}<t$ 的位置,记为 i_{end},i_{end} 即为波瓣的终点,在 i_{start}~i_{end} 范围内与波峰对应的边界点即为拐点。当扫描完所有边界像素后,该步骤结束,其中 t 为阈值,t 取决于边界噪声以及需检测的最大拐角角度 α,由式(3)可得

$$t = 3\varepsilon\left(\delta \frac{\pi-\alpha}{2\pi/k}\right) \tag{4}$$

式中,系数 $\delta(0<\delta\leqslant 1)$ 决定了检测出的最大拐角与 α 的近似程度,ε 为边界噪声曲率的平均值,一般 $1\leqslant\varepsilon\leqslant 3$,一般情况下,$\alpha$ 取 $5\pi/6$,即 $t=k\varepsilon\delta$,而 $\varepsilon\delta\approx 1$;

(5)过滤伪拐点　若步骤(4)检测出的两相邻拐点间曲线距离小于 d,则选曲率较大的一点为拐点,而另一点为伪拐点;

(6)判断拐点的凸凹性　不失一般性,设二值图像中黑色像素值为 0,白色像素值为 1,若

$$\sum_{v=1}^{d/2} \mathrm{mid}(p_{u-v}, p_{u+v}) \leqslant \frac{d}{4} \tag{5}$$

则该点为凸拐点,否则为凹拐点。式(5)中,u 为拐点在边界像素中的索引值;$\mathrm{mid}(p_{u-v},p_{u+v})$ 为像素 p_{u-v}、p_{u+v} 中点的像素值。

2.4　复杂度分析

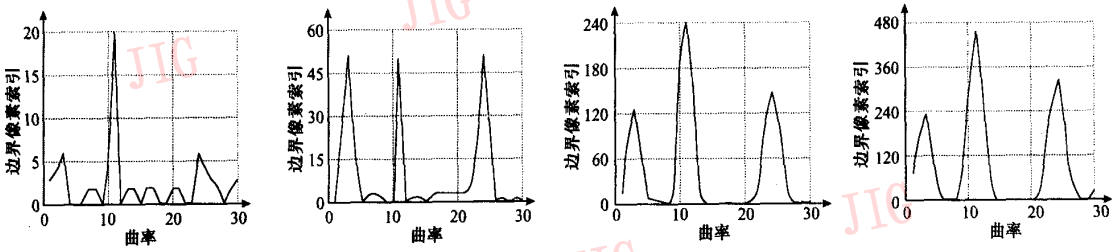
从以上算法描述可以看出,本算法的复杂度为 $O(n)$,n 为图像中物体边界上的像素数目。虽然大多数基于边界点的拐点检测算法的复杂度的量级均为 $O(n)$,但运算实现的时间开销不同。本算法中,对于每一边界点主要进行式(1)~式(3)的运算,约

为 6 次加法和 1 次乘法;FD 算法中,每一边界点至少进行 2 次开方运算(或对数运算)、3 次乘法运算和 5 次加法运算[5];RJ 算法中,每一边界点根据余弦定理平均计算 7 ~ 10 次[3]。BT 算法和 RW 算法在每一边界点的运算量又分别约为 FD 算法和 BT 算法的 3 倍和 10 倍[4,6]。

另一方面,当 k 从 k_1 增大到 k_2 时,本算法增加的时间开销主要分布在以下两个部分:(1)用 k 邻域链码表示边界像素,这部分将增加约 $n(k_2-k_1)/2$ 次判断;(2)利用式(5)判断拐点的凸凹性时,该部分增加约 $s(k_2-k_1)/4$ 次加法运算,s 为拐点的数目。一般来说,由于 $s \ll n$,因此这部分增加的开销可忽略不计,而且随着 k 值的有限增加,整个运行时间并不会急剧增长,这是因为:(1)不论是判断语句,还是加法运算,相对于乘法运算而言,在较少的机器周期内便能完成;(2)实验证明,这些增长时间开销在 k 值有限增加的情况下,占总运算时间的比例并不大;(3)对于一般获得的实际图像的 k 值取 40 已足够平抑边界噪声。因此,理论上本算法的实时性优于上述 4 种算法。

3 实验结果

图 3(a) ~ 3(d)分别为利用式(3)计算得到的图 2 封闭边界的 8 邻域链码 ~ 32 邻域链码曲率直方图。由于 8 邻域中的方向量化等级较少,因而使得图 3(a)中拐点的最低曲率与边界噪声的曲率仅相差 4。在这种情况下,若选择一阈值来探测曲率的局部峰值,则该阈值将因物体的旋转、尺寸以及拐角角度的变化而变得极不稳定,从而使算法缺乏稳定性,甚至会由于边界噪声的干扰而不能正确检测出拐点。图 3(b)中,相对于前者,16 邻域链码由于提高了方向量化的精度,使得拐点的曲率与边界噪声曲率的差别比较明显,且阈值 t 也较稳定,从而提高了算法的稳定性。该实验说明,随着 k 值的增大,不仅拐点处的曲率得到了有效地提升,边界噪声也得到有效抑制。与此同时,曲率直方图上各波瓣的底部随着 k 值增加而逐渐变宽,当 $k \geq 4d$ 时,由于将出现波瓣底部相交的情况,这将会抬高波瓣的起始值,并会干扰检测结果,因此 k 值不能无限大。



(a) $k=8$ (b) $k=16$ (c) $k=24$ (d) $k=32$

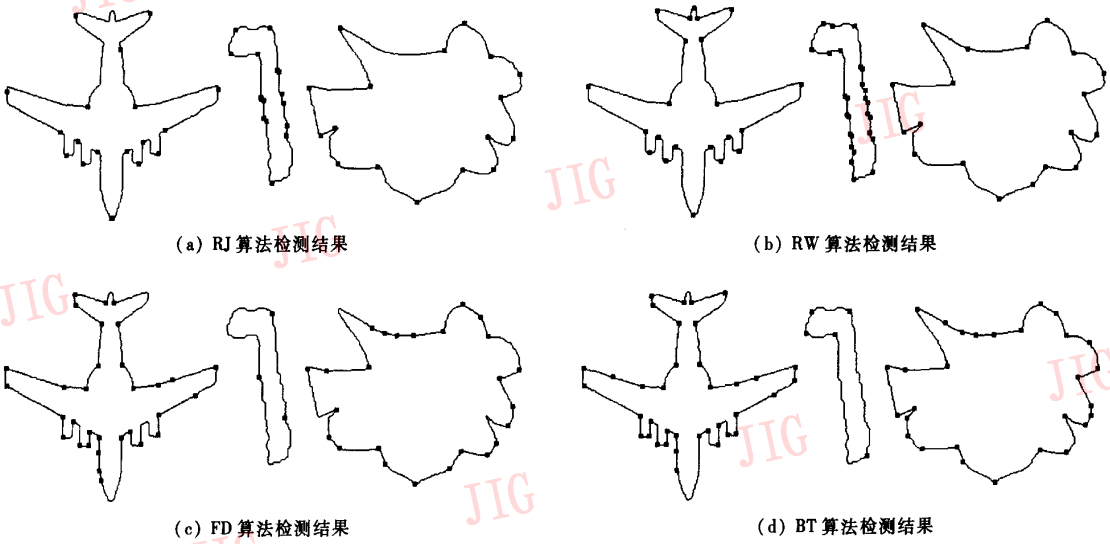
图 3 曲率直方图

Fig. 3 Curvature histogram

为了比较本算法的性能,本文用 C 语言实现了上述算法及另外 4 种拐点检测算法(RJ 算法[3]、RW 算法[4]、FD 算法[5]和 BT 算法[6])。测试图像(如图 4 所示,尺寸为 650pixels × 250pixels)来自文献[7,8]。为了使实验结果具有可比性,每一算法在拐点检测时均使用其缺省参数(即检测结果整体效果最好的参数),而并非对图 4 中每一物体均做参数调整。表 1 为各算法使用的参数以及处理时间。其中,实验环境 A 为 Windows2000、赛扬 800、512M RAM;环境 B 为 eLinux BSP0.3.4、Motorola MC9328MX1、64M RAM;各算法在相应环境中的处理时间为对图 4 循环处理 100 次后得到的平均时间。图 5(a) ~ 5(d)分别为跟踪得到的物体边界以及用上述 4 种算法检测的结果,检测出的拐点用“■”标识。图 6(a)和 6(b)分别为本文算法使用与 $k=32$ 对应的缺省阈值($t=30$)和缺省阈值增加 100% 后($t=60$)检测的结果,其中凹拐点用“×”标识,凸拐点用“■”标识。

图 4 测试图像

Fig. 4 Testing image

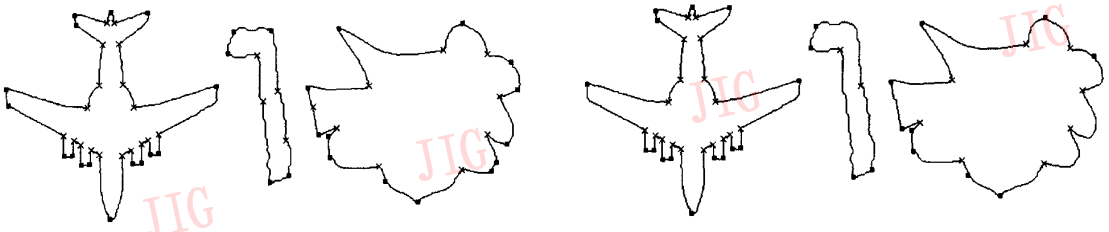


(a) RJ 算法检测结果　(b) RW 算法检测结果

(c) FD 算法检测结果　(d) BT 算法检测结果

图 5　4 种算法拐点检测结果

Fig. 5　Testing results of 4 corner detection algorithms



(a) $t=30$　(b) $t=60$

图 6　本文算法检测结果

Fig. 6　Testing results of the new algorithm

表 1　各算法参数及处理时间

Tab. 1　Parameters and processing time of algorithms

算法	参数	检测出的拐点数	处理时间(ms)	
			环境 A	环境 B
RJ	$n=20$	52	6.00	775.00
RW	$n=20$	68	68.30	17890.00
FD	$s=7,k=1200$	59	5.10	999.00
BT	$s=7,k=1200$ $k_1=4,k_2=7$	70	14.20	3680.00
本文算法	$d=8,t=30$	64	4.27	43.00
	$d=8,t=60$	55	4.26	43.10

从实验结果可以看出本文算法的以下几个优点:(1)在环境 A 中,RJ 算法与 FD 算法的处理时间虽与本文算法接近,但其错检、漏检的拐点较多;RW 算法在检测质量上虽然比 RJ 算法有所提高,但该算法在 4 种算法中实时性最差。在环境 B 中,4 种算法处理时间增长的幅度远远大于本算法,其原因在于,MC9328MX1 虽基于 ARM9,但不支持浮点运算,这也是大多数嵌入式计算环境的共有特征,另外,其他 4 种算法中均包含大量浮点运算,而本文算法所有运算均可基于整数实现;(2)比较图 6(a)和 6(b)可看出,在阈值 t 增加 100% 后,检测结果仅稍有变化,这在上述 4 种算法中是做不到的;(3)本文算法可正确区分出拐点的凸凹性。

表 2 为本文算法在不同 k 邻域中,对图 4 进行拐点检测时的处理时间比较。其中环境 A 和环境 B 处理时间的获得方式均与表 1 相同;增幅为算法在各 $k(k>8)$ 邻域内环境 A 和环境 B 中相对于 8 邻域处理时间增幅的平均值。从该实验结果可看出本文算法的以下几个特点:(1)k 值从 8 增长到 40 的过程中,处理时间的相对增幅不超过 50%,绝对增幅在环境 A 中不超过 1ms,在环境 B 中不超过 18 ms;(2)k 值每增加 8,处理时间增幅不超过 12%,这也证实了本文在 2.4 节中的第 2 点论断;(3)即便是在 $k=40$ 的情况下,本文算法的实时性也优于前面 4 种算法;(4)算法以较少的处理时间为代价,得到

表 2 本文算法在不同 k 邻域中的处理时间

Tab. 2 Processing time of the new algorithm using different k value

参数			检测出的拐点数	处理时间(ms)		
d	k	t		环境 A	环境 B	增幅(%)
2	8	4	33	3.52	30.60	-
4	16	12	55	3.73	33.70	8.0
6	24	22	60	4.03	37.90	19.4
8	32	30	64	4.27	43.00	30.9
10	40	40	66	4.49	47.70	41.7

了稳定、准确的检测结果。

4 结 论

与传统基于边界链码的拐角检测算法不同,由于本算法通过采用 k 邻域链码充分提高了边界各点方向量化的精度,且增加了拐点曲率与边界噪声曲率的差别,从而提高了检测结果的准确性和稳定性,并且所有运算均可基于整数实现。目前该算法已成功应用于二维条码识别系统中,大量测试表明,本算法不仅准确性较高,而且算法简单、实时性强,适合于嵌入式计算环境。

参考文献(References)

1 Smith S M, Brady J M. SUSAN—A new approach to low level image processing[J]. International Journal of Computer Vision, 1997, **23**(1): 45 ~ 78.

2 Li Hua, Liu Wenyu, Zhu Yaoting, *et al.* A uniform model of corner detection based on morphology[J]. Journal of Image and Graphics, 2002, **7**(6): 543 ~ 547. [李华, 刘文予, 朱耀庭等. 基于形态学的快速拐点检测统一模型[J]. 中国图象图形学报, 2002, **7**(6): 543 ~ 547.]

3 Rosenfeld A, Johnston E. Angle detection on digital curves[J]. IEEE Transactions on Computers, 1973, **22**(9): 875 ~ 878.

4 Rosenfeld A, Weszka J S. An improved method of angle detection on digital curves[J]. IEEE Transactions on Computers, 1975, **24**(9): 940 ~ 941.

5 Freeman H, Davis L S. A corner finding algorithm for chain code curves[J]. IEEE Transactions on Computers, 1977, **26**(3): 297 ~ 303.

6 Beus H L, Tiu S S H. An improved corner detection algorithm based on chain-coded plane curves[J]. Pattern Recognition, 1987, **20**(3): 291 ~ 296.

7 Liu H C, Srinath M D. Corner detection from chain-code[J]. Pattern Recognition, 1990, **23**(1): 51 ~ 68.

8 Chetveerikov D, Zsolt Szabó. Detection of high curvature points in planar curves [EB/OL]. http://visual.ipan.sztaki.hu/corner/mode8.html. 1999.