一种面向 FPGA 的快速 Hough 变换

商尔科,李 健,安向京
SHANG Er-ke,LI Jian,AN Xiang-jing

国防科技大学 自动化研究所,长沙 410073
Institute of Automation,National University of Defense Technology,Changsha 410073,China
E-mail:erke1984@qq.com

SHANG Er-ke,LI Jian,AN Xiang-jing.Fast Hough transform for FPGA-based applications.Computer Engineering and Applications,2010,46(7):72-75.

Abstract: This paper presents a FPGA based fast Hough transform for line detection.The input edge image is divided into several directions by using classified filter,so every direction can be processed in parallel.And for each direction,a pipelined structure is used for line extraction.A parameter storage and search strategy is proposed,with which 2-step histogram search strategy is introduced in the purpose of reducing the storage requirement.Experiments show that the algorithm is efficient,low storage requirement and real time.

Key words: Hough transform;Field Programmable Gate Arrays(FPGA);classified filter;pipelined structure

摘 要:在 FPGA 上设计并实现了一种用于直线检测的快速 Hough 变换方法。使用分类滤波器把直线目标分成多个方向,使多个方向上的运算在空间上实现了并行处理;在每个方向上,设计实现了一种用于 Hough 变换的流水线处理结构;提出了一种基于直方图统计的两阶段搜索算法。大量的实验验证了提出的 Hough 变换实现方法的可行性,结果证明该方法占用空间少,实时性高。

关键词:霍夫变换;可编程逻辑门阵列;分类滤波器;流水线结构
DOI:10.3778/j.issn.1002-8331.2010.07.022 **文章编号:**1002-8331(2010)07-0072-04 **文献标识码:**A **中图分类号:**TP391

1 引言

Hough 变换由于其良好的鲁棒性和抗干扰性而被广泛地用于直线检测。但是标准 Hough 变换[1]的缺点也很明显:需要巨大的运算量和存储空间。因此 Hough 变换的应用受到了很大的限制。近年来,研究者提出了很多用于降低 Hough 变换运算复杂度和存储空间的改进算法[2-3]。

随着微电子技术的进步,现场可编程逻辑门阵列(Field Programmable Gate Arrays,FPGA)技术由于其设计灵活性、大规模并行运算等特点,为 Hough 变换的实现提供了一种新的思路。

根据 FPGA 的特点,提出了一种面向 FPGA 的快速 Hough 变换方法。首先,使用分类滤波器把边缘图像中的直线目标分成多个方向,降低了每个方向上的特征点总数,并且各个方向上的运算在空间上实现了并行处理;然后,在每个方向上设计了一种用于 Hough 变换的流水线处理结构,极大降低了运算时间;最后,提出了一种新的参数存储与搜索方法,提高了搜索效率,减少了存储空间。实验证明了该方法的可行性和快速性。

2 Hough 变换的原理

标准 Hough 变换(Standard Hough Transform,SHT)的基本思想[1]是利用点-线的对偶性原理,把直角坐标系中的每一个特征点,映射到参数空间的累加阵列的多个单元中,再统计各个单元的计数以检测出极大值,确定是否存在直线并获得直线参数。

标准 Hough 变换算法具有很大的盲目性,需要巨大的存储空间和运算量[4],难以满足存储空间较小、实时性要求较高场合的应用。

2.1 随机 Hough 变换

文献[2]提出了一种随机 Hough 变换(Randomized Hough Transform,RHT)方法,引入了特征点的随机采样、图像空间到参数空间的“多对一”的机制,有效地降低了运算复杂度和存储空间。

随机 Hough 变换的基本思想为:首先存储所有的图像特征点的位置信息(x,y),然后随机地选取特征点对(x_i,y_i)和(x_j,y_j),根据公式(1)和公式(2)来计算参数空间的(θ,ρ),最后对参数空间的值进行投票累加得到可能直线的位置。

$$\theta_{ij}=\text{arctg}\left(\frac{x_j-x_i}{y_j-y_i}\right) \tag{1}$$

$$\rho_{ij}=x_i\times\cos(\theta_{ij})+y_i\times\sin(\theta_{ij}) \tag{2}$$

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No.90820302,No.90820015)。
作者简介:商尔科(1984-),男,硕士生,主要研究方向为图像处理,嵌入式系统开发等;李健(1981-),男,博士生,主要研究方向为模式识别与人工智能等;安向京(1973-),男,副教授,主要研究为模式识别与人工智能等。
收稿日期:2009-07-16 **修回日期:**2009-08-31

随机 Hough 变换采用图像空间到参数空间的“多对一”映射思想，避免了标准 Hough 变换的盲目性，有效地降低了算法复杂度和存储空间。但随机 Hough 变换也有其不足之处，即所需的存储空间和运算时间会随特征点的增多而呈指数级增长，导致变换实时性较差[5]。

2.2　分类 Hough 变换

文献[3]提出了一种分类 Hough 变换(Classified Hough Transform，CHT)方法。分类 Hough 变换是对随机 Hough 变换的一种改进，它通过引进两种机制：分类机制和等距取点机制，来取代以往的全域搜索和穷举法或随机采样，从而进一步降低了算法的运算量。其主要思想是根据目标要求，预先构造一个分类器，把目标按照一定的标准分成几类，每一类都有他们的共性，因此针对某一类目标就可以大大降低其总的特征点个数，从而降低运算量和存储空间。

分类 Hough 变换的优点是特征点总量减少，有效采样率高，计算量小，算法效率比随机 Hough 变换进一步提高。但是也存在着明显的缺点：采用了等距取点配对机制，削弱了对短线段的检测能力[5]。

3　面向 FPGA 的快速 Hough 变换

为了解决上述各种 Hough 变换的缺点，进一步降低存储空间，提高其实时性，提出了一种面向 FPGA 的快速 Hough 变换方法(如图 1 所示)，其基本思想为：首先设计 n 个方向滤波器对原始边缘图像进行滤波，得到 n 幅不同方向的边缘图像，使得每幅边缘图像中的特征点总数大为减少，从而降低了运算量；与此同时，通过将每个方向上的 Hough 变换设计成一个独立模块，可以实现多个方向的 Hough 变换在空间上的并行处理。其次，通过存储参数空间的(θ,ρ)来代替存储图像空间的特征点信息，可以将每个方向上 Hough 变换设计成一个流水线结构，从而实现了 Hough 变换在时间上的并行处理。最后，提出了一种基于直方图统计的两阶段搜索算法，该算法将传统 Hough 变换中所需的二维空间的搜索转为两次一维搜索，不仅提高了算法效率，更主要的是将寻优过程中所需的二维存储空间降为一维存储空间，有效减轻了硬件的资源占用。

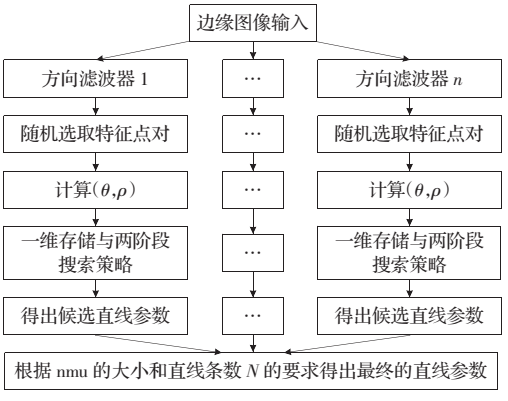

图 1　快速 Hough 变换流程图

3.1　方向滤波器

借鉴了分类 Hough 变换的分类思想，并结合 FPGA 的空间并行运算优点，构造了 A、B、C 三个不同斜率的方向滤波器，可同时对正斜率直线、接近 0 斜率直线和负斜率直线分别进行滤波，使这三类直线的 Hough 变换在空间上同时进行(如图 2 所示)。

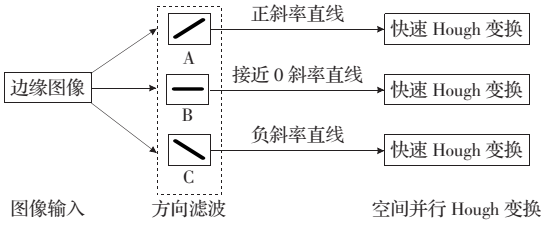

图 2　三类直线快速 Hough 变换的空间并行处理示意图

方向滤波器采用 3×5 的滤波模板实现。对于某个像素点 A_{23}，其周围像素点如下：

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} & A_{15} \\ A_{21} & A_{22} & A_{23} & A_{24} & A_{25} \\ A_{31} & A_{32} & A_{33} & A_{34} & A_{35} \end{bmatrix}$$

以正斜率方向滤波器 A 为例，设计的滤波模板如下：

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

以此为基础，设计的滤波公式为：

$$A_{23}=(A_{13}\cup A_{14}\cup A_{15})\cap(A_{22}\cup A_{23}\cup A_{24})\cap (A_{31}\cup A_{32}\cup A_{33})\cap A_{23} \quad (3)$$

公式(3)表明，当模板相邻三行中，每行都有特征点时输出中心点 A_{23} 的值，否则认为 A_{23} 不可能是正斜率直线上的特征点。通过分析可知，滤波器 A 对直线的角度覆盖范围为(26°，90°]。

方向滤波器 B、C 的工作原理与 A 类似，角度覆盖范围分别为(0°，26°]∪(154°，180°]和(90°，154°]。

据此，滤波器 A、B、C 把原边缘图像中可能存在的直线有效地分成了三类，降低了后续处理的复杂度。

3.2　流水线结构

为了提高运算效率，通过存储参数空间的(θ,ρ)来代替存储图像空间的特征点信息，改变了随机 Hough 变换中先存储一整幅图像的特征点信息然后再进行 Hough 处理的工作模式，使 Hough 变换可以被设计成一个流水线结构。边缘图像通过方向滤波器后得到特征点的位置信息，然后随机地选取特征点对进行(θ,ρ)的运算(如图 3 所示)。

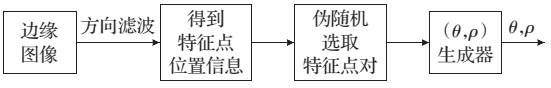

图 3　快速 Hough 变换的流水线结构图

3.2.1　伪随机特征点对的选取

在随机 Hough 变换中，需要把一整幅图像的特征点信息存储下来，然后再对存储的点进行运算[2]，这样既需要额外的存储空间，又无法采用流水线结构在时间上并行处理，因此运算效率不高。

事实上，特征点的位置信息只是在用来计算(θ,ρ)的随机选择过程中被有限次地使用。鉴于此，提出并实现了一种通过多级缓存的流水线结构实现局部范围内的多步长伪随机取点的方法，对一定范围内的多个固定间隔的特征点按一定的策略进行选取，既保持了随机 Hough 变换随机取点的多样性，又继承了分类 Hough 变换定步长取点降低计算量的优势。多步长伪随机取点的示意图如图 4 所示，特征点信息通过 n 级不同深度的缓存单元进行流水传递，而且每级缓存都可以并行输出其

特征点信息,因此在任一时刻,通过自行设计的伪随机采样策略，就可以从这 n+1 个特征点中随机选取两个点作为 Hough 变换的采样点对。

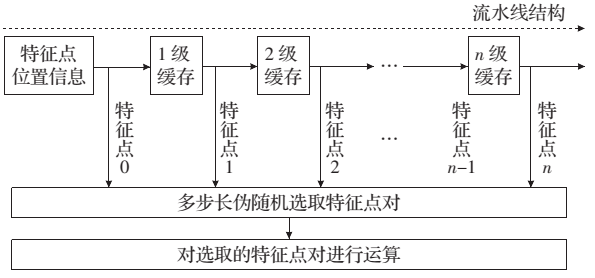

图 4　基于流水线结构的多步长伪随机取点示意图

3.2.2　(θ,ρ)生成器

在 Hough 变换中,最关键的运算是根据公式(1)和公式(2)来计算参数空间的(θ,ρ)。由于它涉及的三角函数运算需要多个时钟周期才能完成,因此,(θ,ρ)的计算速度是 Hough 变换效率的瓶颈之一[7]。为了解决这一问题,设计了一种流水线结构的(θ,ρ)生成器,其基本思想为:在特征点对流水线输入的情况下,通过一系列流水线运算单元,使得每一对(θ,ρ)的计算都可以在一个时钟内完成,具体如图 5 所示。

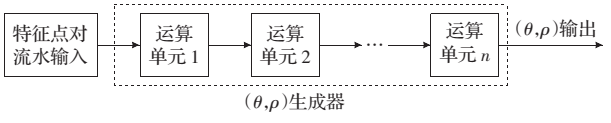

图 5　基于流水线结构的(θ,ρ)生成器示意图

3.3　基于直方图统计的两阶段搜索算法

3.3.1　随机 Hough 变换的二维存储与搜索方法

随机 Hough 变换是通过一个二维存储空间来对可能直线上的点进行计数，之后再对此二维存储空间进行遍历搜索,这种方法不仅效率低,而且需要很大的存储空间。以一幅 762×400 的图像为例,需要建立的二维存储空间为(180,861),假设每个存储单元的位数为 10 位,则它所需的存储空间为 193 725 Byte,这对于单片 FPGA 来说,实现起来比较困难,资源浪费也很严重。实验表明,通常情况下其资源利用率还不到 1%。

3.3.2　提出的两阶段搜索算法

为了降低存储空间,提高算法效率,提出了一种基于直方图统计的两阶段搜索算法(如图 6 所示)。其基本思想为:当在某个方向上需要寻找 N 条直线时，首先对每一次运算得到的(θ,ρ)进行一维存储,同时在方向 ρ 上做一维的直方图统计,当图像输入结束后,就可以得到所有(θ,ρ)的一维存储信息和 ρ 方向上的 N 个极大值点 $\rho_1,\rho_2,\cdots,\rho_N$;然后根据 ρ_i 的值,再对一维存储的(θ,ρ)进行 θ 方向上的一维直方图统计,就可以在每一个 ρ_i 对应的 θ 方向上得到 N 组极大的$(\theta_j,\rho_i,num_\theta)$参数;最后根据 num_θ 的大小来选择最有可能存在的 N 条直线信息(θ,ρ)。

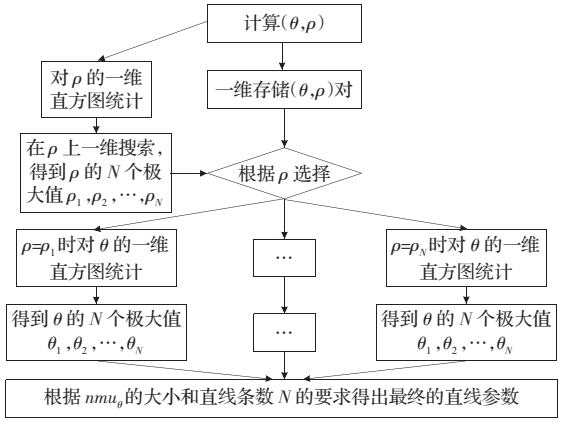

图 6　基于直方图统计的两阶段搜索算法示意图

为简单起见,以图 7 中的直线为例来说明此方法。表 1 给出了该直线在二维参数空间(θ,ρ)的局部信息。表 2 是对二维参数(θ,ρ)在ρ 方向进行一维直方图统计所得到的局部信息表,表 3 是根据的 ρ 极值对二维参数(θ,ρ)在 θ 方向进行一维直方图统计所得到的局部信息表。

图 7　简单直线分布示意图

表 1　图 7 中直线在参数空间(θ,ρ)的局部信息

ρ	θ/(°)				
	43	44	45	46	47
50	0	0	0	0	0
51	0	0	10	0	0
52	0	10	50	10	0
53	0	0	10	0	0
54	0	0	0	0	0

表 2　基于ρ 的一维直方图统计局部信息表

ρ	50	51	52	53	54
num_ρ	0	10	70	10	0

表 3　基于 θ 的一维直方图统计局部信息表(ρ=52)

θ/(°)	43	44	45	46	47
num_θ	0	10	50	10	0

根据提出的方法,N=1,这样表 1 中的最大值(45°,52)可通过先搜索表 2 中 ρ 方向上的一维直方图得到最大值 ρ=52,然后再从表 3 中 ρ=52 时 θ 方向上的一维直方图找出 θ 的最大值 θ=45°,从而得到参数组(θ=45°,ρ=52,num_θ=50)。

当图像中有 K 条直线并且要找 N 条($N\leqslant K$)直线时,上述方法同样适合。

3.3.3　特殊情况分析

上述方法对于一般的直线分布是可行的,但当直线的分布出现比较特殊的情况时，如出现同圆的多条切线时就不够完善。尤其是当 N=1 时采用该方法寻找图像中的最长直线,可能会被多条短直线的信息淹没而导致找到的直线不是最长的直线。如图 8 所示,由于圆弧和两短直线都有相同的 ρ,因此当只要寻找一条直线时,在 ρ 上的一维直方图中搜索到的极大值就有可能是圆弧和两条短直线对应的 ρ 值,而不是图中的长直线所对应的 ρ 值。

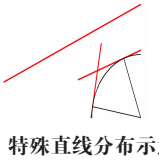

图 8　特殊直线分布示意图

解决特殊分布的一种尝试是增加寻找可能直线的条数。当需要寻找的直线为 N 时，在 ρ 方向的一维直方图中寻找 M 个极大值点(M>N),然后对每一个 ρ 值对应的 θ 直方图中寻找 M 个极大值,共得到 M×M 组可能的直线参数,然后根据 num 的

大小来选择最有可能的 N 条直线的位置。这种方法的好处是可以解决长直线被淹没的问题,缺点是由于 FPGA 硬件实现缺乏灵活性,不能动态地改变 M 值,而且 M 值的选择比较困难,反映了运算精度和存储空间的矛盾。

3.3.4　资源分析与比较

随机 Hough 变换的二维存储资源分析如 3.3.1 节所述,对于一幅 762×400 的图像,需要存储空间 193 725 Byte,所提方法的资源分析如下:

方向滤波器个数=3;每个(θ,ρ)一维存储模块的深度为 4 096,位置信息用 10 位表示,则所用资源为 5 120 Byte;每个 θ 一维直方图存储的深度为 60,用 10 位来计数,所用资源为 75 Byte;每个 ρ 一维直方图存储的深度为 861,用 10 位来计数,所用资源为 1 076.25 Byte;N 表示每个方向上需要寻找的直线条数。表 4 给出了该方法所需的资源以及与传统二维存储之间的资源比较情况。

表 4　资源分析与比较

N	(θ,ρ)存储/个	ρ 存储/个	θ 存储/个	所需资源/Byte	与二维存储之比
1	3	3	3	18 814	0.097 1
2	3	3	6	19 039	0.098 3
3	3	3	9	19 264	0.099 4
4	3	3	12	19 489	0.100 6
…	…	…	…	…	…
n	3	3	3×N	(18 589+225×n)	(18 589+225×n):193 725

4　实验结果与分析

实验采用了 Xilinx 公司的 Vertex2 系列 100 万门 FPGA,使用的软件是 ISE 9.2i 版本。实验是在使用 FPGA 实现摄像机数据采集、图像中值滤波和边缘二值化的基础上实现的,时钟频率为 33 MHz,图像大小为 400×762 的 8 位灰度图,实验中只做了图 2 中的 A 与 C 两个方向模板滤波,直线检测参数 N=2,即每个模板方向上寻找两条直线,寻找过程按照 3.3.2 节的搜索算法实现,经过计算后得到可能直线的参数(θ,ρ,num_θ),根据 num_θ的值判断直线存在的可信度,认为是直线后把直线参数发送给 PC 机,在 PC 机上用直线在原图像中标示出直线的位置。图 9 分别为原图、LoG 算子边缘图、A 模板滤波图、C 模板滤波图和原图中标记直线位置图。

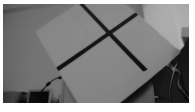

图 9(a) 原图

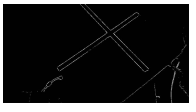

图 9(b) 边缘增强图

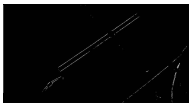

图 9(c) A 模板滤波效果图

图 9(d) C 模板滤波效果图

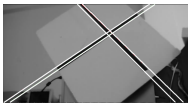

图 9(e) 直线位置效果图

针对 3.3.3 节中图 8 的特殊直线分布情况,做了专门的实验。图 10 三个图分别为 N=1 时寻找一条最长的直线,结果被短直线的信息淹没导致寻找到的直线不是最长的直线;N=2 时寻找两条直线的情况;采用了 3.3.3 节中提出的增加寻找可能直线的条数来解决寻找一条直线时长直线被淹没的问题,取 M=2,N=1。实验表明,在一幅图像中拥有多条相同 ρ 的短直线和圆弧时,通过寻找更多的候选直线来解决长直线被淹没的方法是可行的。

图 10　实验结果图

算法应用于实际道路中的车道线检测,其效果图如图 11 所示,N=1,即在每个滤波方向上检测一条最长的车道线位置。

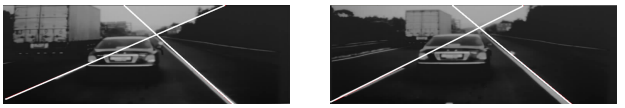

图 11　实际场景的应用

使用的时间和资源分析如表 5 所示。

表 5　时间资源使用表

功能模块		IA	IA+MF	IA+LoG	IA+MF+LoG	IA+MF+LoG +HT
时间/ms		9.24	9.28	9.28	9.32	9.42(9.452)
资源/个	Total LUTs	253	558	404	785	6 148
	Flip Flop	239	444	383	663	4 258
	Block RAM	0	2	2	4	18
	MULT18×18 s	0	0	0	0	4
	EG	5 538	148 339	144 062	282 729	1 326 082

实验在单片 FPGA 中完成,没有使用外部的存储单元,所开辟的一维存储资源如 3.3.4 节所示,当图像中所有特征点运算结束或者(θ,ρ)存储空间满时开始参数空间的搜索,因此所列的时间为算法所需的平均时间,括号中的时间为当存储空间满时理论上需要的最大计算时间。表 5 中 IA 表示数据采集(Image Acquire),MF 表示中值滤波过程(Median Filter),LoG 表示 LoG 边缘检测过程,HT 表示 Hough Transform 变换过程。EG 表示等价门数(Total equivalent gate count for design)。

文献[6]也在单 FPGA 上实现了 Hough 变换的功能,它使用的 FPGA 型号为 Xilinx 公司的 XC4VLX200,实现文献[1]中提到的标准 Hough 变换,处理的图像大小为 640×480,从资源上和本实验做了比较,如表 6 所示。

表 6　与文献[6]的资源比较表

		文献[7]	本实验
	图像大小	640×480	762×400
资源/个	4 input LUTs	13 517	5 189
	Flip Flop	17 713	4 258
	Occupied Slices	13 793	3 777
	Block RAM	246	18
	MULT18×18s 或 DSP48	60	4
	Total equivalent gate count for design	16 452 974	1 326 082

从表 6 可以看出,与标准 Hough 变换相比,该方法极大地降低了所需的存储空间。从运算时间上分析,快速 Hough 变换只带来了 0.1 ms 左右的额外运算时间,因此也完全满足了实时性的要求。

5　结束语

借鉴了随机 Hough 变换“多对一”映射、分类 Hough 变换构造分类器和定步长采样的思想,根据 FPGA 可以空间并行运算和流水线并行运算的特点,设计了一种面向 FPGA 的快速

(下转 95 页)